Házi feladat

Programozás alapjai 2.

Végleges

Szombati Olivér István
P37PLU

2021. május 14.

Feladatspecifikáció:

Feladat:

Kő-papír-olló játék

Tervezzen objektummodellt a kő-papír-olló játék modellezéséhez! Célunk, hogy a különböző stratégiával játszó játékosokat összesorsolva megállapítsuk a legjobb stratégiát, ha van ilyen. A modellben legyenek "Játékos" objektumok, melyek egy "Napló" objektum felügyeletével játszanak. Ez utóbbi gyűjti a statisztikát.

Demonstrálja a működést külön modulként fordított tesztprogrammal! A játék állását nem kell grafikusan megjeleníteni, elegendő csak karakteresen, a legegyszerűbb formában! A megoldáshoz ne használjon STL tárolót!

Feladatspecifikáció:

A feladat a kő-papír-olló nevezetű játék objektum orientált megvalósítása. A program vezérli két játékos[1] mérkőzését a megszokott szabályok szerint[2].

A játékosok statisztikáit egy Napló objektum fogja tárolni, ami a legjobb 10 játékost és azoknak a statisztikáit fogja majd kiírni[3]. Azok a legjobb játékosok, akiknek a legnagyobb győzelmi szériája van.

A Napló képes lesz:

- Eltárolni a 100 legjobb ember[4] nevét, statisztikáját.
- Kiírni a 10 legjobb játékos nevét, statisztikáját és győzelmi szériájának számát.
- Egy verseny után elmenteni az új vagy már meglévő játékosok adatait.

[1] A játékosok neveit be lehet állítani, a statisztikájukat elmenti a játék.
[2] Mindketten egyszerre felmutatnak egyet a 3 lehetséges tárgy közül, a kő legyőzi az ollót, az olló legyőzi a papírt és a papír legyőzi a követ.
[3] Viszont ez nem azt jelenti, hogy 10 embernél többet nem tárolhat.
[4] Statisztikák vizsgálatához elegendőnek tekintettem ennyit.

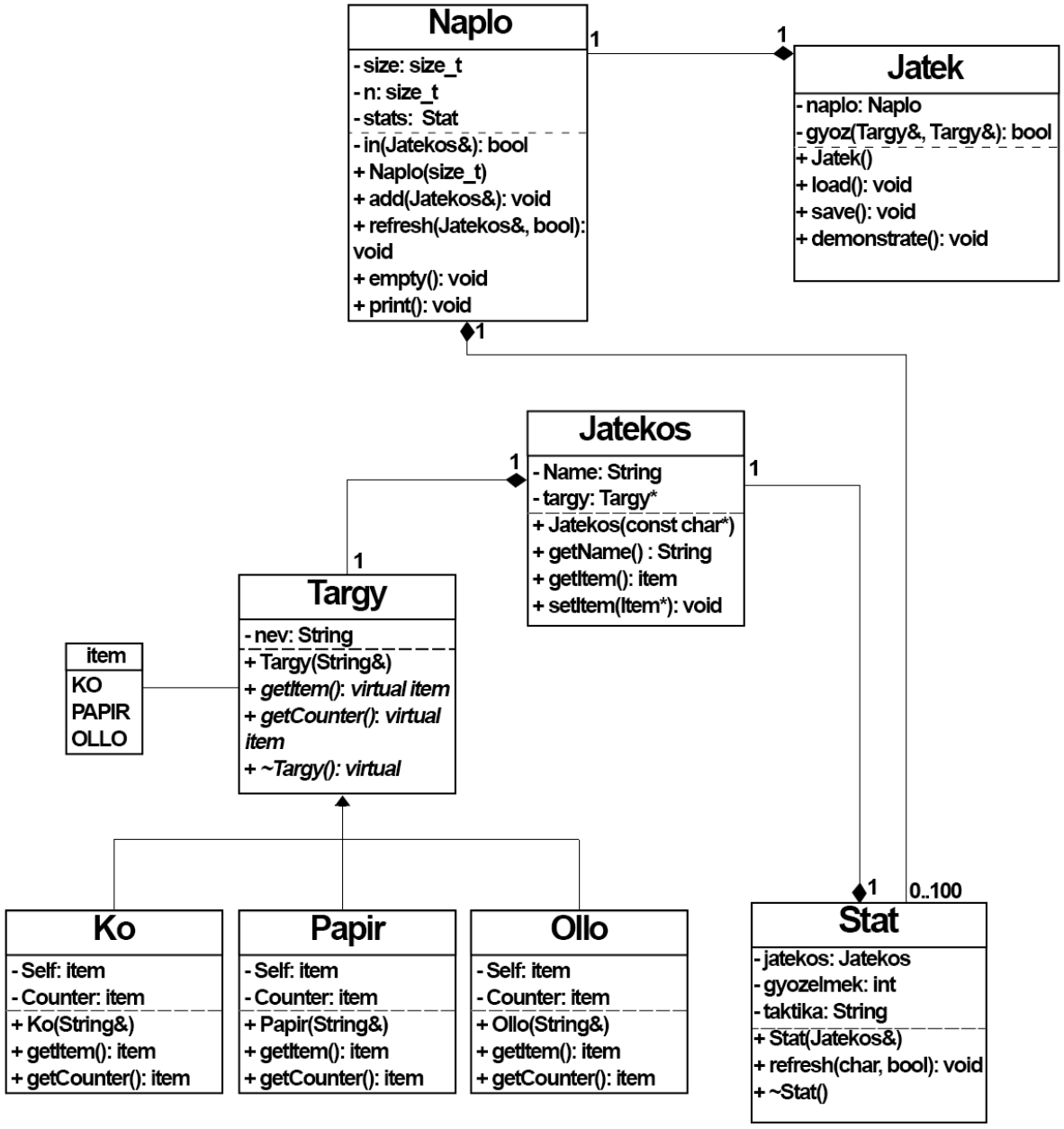
A Játék objektum vezérli két játékos versenyét. A következő műveleteket végzi el egy verseny alatt:

- Kér két nevet (ha még nem szerepel az adatbázisban, hozzáadja őket, ha igen, akkor bővíti a statisztikájukat)
- A program lényege az OO megvalósítás, ezért a két játékos „bot" lesz, azaz a számítógép véletlenszerűen választ nekik tárgyat.
- Eldönti, hogy melyik játékos tárgya erősebb, és ennek megfelelően kiosztja a győztesnek a győzelmi pontot, a vesztesét nullázza.
- A naplót frissíti az eredményektől függően (sorba rendezés, stratégia kiírása[5])

[5] A stratégia kiírása úgy zajlik, hogy a játékos egy új győzelmétől számítva a győztes körök választott tárgyainak kezdőbetűit kiírja egymás után. pl. *Jatekos1 – KKPOKOPP (8)*

Terv:

Osztálydiagram:



Megjegyzések:

- A nevek a későbbiekben minimálisan, de eltérhetnek az ábrán jelöltektől (pl. Self ~ SajatTargy)
- Bár az ábrán úgy tűnik, hogy a Targy osztály alosztályai ugyanazok, valójában a konstruktoruk egy-egy fontos lépésben el fog térni

Fontosabb algoritmusok leírása:

(**Jatek**) void demonstrate():

- Bekéri az első játékos nevét
- HA a játékos NINCS benne a naplóban:
  Naplóhoz hozzáadja a játékost
- Bekéri a második játékos nevét
- HA a játékos NINCS benne a naplóban:
  Naplóhoz hozzáadja a játékost
- Választ egy véletlenszerű tárgyat
- Első játékosnak beállítja a tárgyat
- Választ egy véletlenszerű tárgyat
- Második játékosnak beállítja a tárgyat
- HA az első játékos nyer:
  Frissíti a naplót:
  refresh(Első játékos, igaz)
  refresh(Második játékos, hamis)
- EGYÉBKÉNT:
  Frissíti a naplót:
  refresh(Első játékos, hamis)
  refresh(Második játékos, igaz)

(**Naplo**) void add(Jatekos&):

- HA n>=size: kivételt dob
- Ciklus a tömb elemein végig
- HA megegyezik valamelyik játékos neve a paraméterként adott játékos nevével:
  STOP (nem tesszük be újra)
- A tömb megfelelő helyére bekerül az új játékos
- n += 1

(**Naplo**) void refresh(Jatekos&, bool):

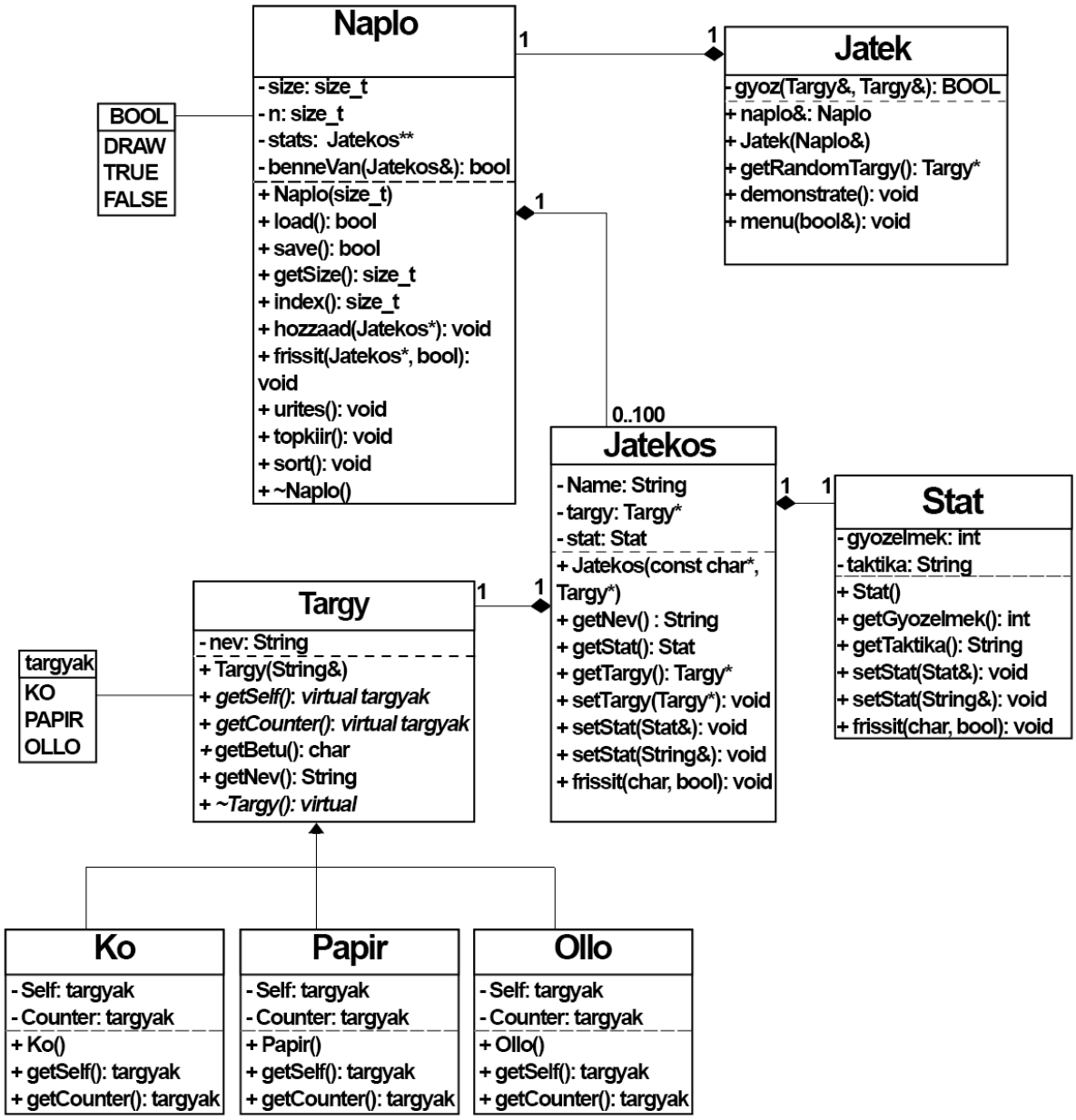
- HA a játékos nincs benne a naplóban:
  MEGPRÓBÁLJA hozzáadni
  HA hiba van: (tele)
  Hibaüzenet írása, return
  EGYÉBKÉNT:
  Játékos győzelmei = 0
  Játékos taktikája = ””
- Frissíti a játékos statisztikáját

(**Stat**) void refresh(char, bool):

- HA a bool IGAZ:
  Játékos győzelmei += 1
  Játékos taktikája += Tárgy betűje
- EGYÉBKÉNT:
  HA a Játékos statisztikája = 0:
  Játékos statisztikája -= 1
  Játékos taktikája += Tárgy betűje
  EGYÉBKÉNT:
  Játékos statisztikája = 0
  Játékos taktikája = ””

Végleges:

Osztálydiagram:



Prog2_NHF

Készítette Doxygen 1.9.1

1. fejezet

Névtérmutató

1.1. Névtérlista

Az összes névtér listája rövid leírásokkal:

2. fejezet

Hierarchikus mutató

2.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

3. fejezet

Osztálymutató

3.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

4. fejezet

Fájlmutató

4.1. Fájllista

Az összes fájl listája rövid leírásokkal:

5. fejezet

Névterek dokumentációja

5.1. gtest_lite névtér-referencia

gtest_lite: a keretrendszer függvényinek és objektumainak névtere

Osztályok

- struct Test

Függvények

- template<typename T1 , typename T2 >
  std::ostream & EXPECT_ (T1 exp, T2 act, bool(*pred)(T1, T2), const char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")

  általános sablon a várt értékhez.
- template<typename T1 , typename T2 >
  std::ostream & EXPECT_ (T1 *exp, T2 *act, bool(*pred)(T1 *, T2 *), const char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")

  pointerre specializált sablon a várt értékhez.
- std::ostream & EXPECTSTR (const char *exp, const char *act, bool(*pred)(const char *, const char *), const char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")
- template<typename T1 , typename T2 >
  bool eq (T1 a, T2 b)
- bool eqstr (const char *a, const char *b)
- bool eqstrcase (const char *a, const char *b)
- template<typename T1 , typename T2 >
  bool ne (T1 a, T2 b)
- bool nestr (const char *a, const char *b)
- template<typename T1 , typename T2 >
  bool le (T1 a, T2 b)
- template<typename T1 , typename T2 >
  bool lt (T1 a, T2 b)
- template<typename T1 , typename T2 >
  bool ge (T1 a, T2 b)
- template<typename T1 , typename T2 >
  bool gt (T1 a, T2 b)
- template<typename T >
  bool almostEQ (T a, T b)

5.1.1. Részletes leírás

gtest_lite: a keretrendszer függvényinek és objektumainak névtere

5.1.2. Függvények dokumentációja

5.1.2.1. almostEQ()

```
template<typename T >
bool gtest_lite::almostEQ (
            T a,
            T b )
```

Segédsablon valós számok összehasonlításához Nem bombabiztos, de nekünk most jó lesz Elméleti hátér: http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm

5.1.2.2. eq()

```
template<typename T1 , typename T2 >
bool gtest_lite::eq (
            T1 a,
            T2 b )
```

segéd sablonok a relációkhoz. azért nem STL (algorithm), mert csak a függvény lehet, hogy menjen a deduckció

5.1.2.3. eqstr()

```
bool gtest_lite::eqstr (
            const char * a,
            const char * b )  [inline]
```

5.1.2.4. eqstrcase()

```
bool gtest_lite::eqstrcase (
            const char * a,
            const char * b )  [inline]
```

5.1.2.5. EXPECT_() [1/2]

```
template<typename T1 , typename T2 >
std::ostream& gtest_lite::EXPECT_ (
            T1 * exp,
            T2 * act,
            bool(*)(T1 *, T2 *) pred,
            const char * file,
            int line,
            const char * expr,
            const char * lhs = "elvart",
            const char * rhs = "aktual" )
```

pointerre specializált sablon a várt értékhez.

5.1.2.6. EXPECT_() [2/2]

```
template<typename T1 , typename T2 >
std::ostream& gtest_lite::EXPECT_ (
            T1 exp,
            T2 act,
            bool(*)(T1, T2) pred,
            const char * file,
            int line,
            const char * expr,
            const char * lhs = "elvart",
            const char * rhs = "aktual" )
```

általános sablon a várt értékhez.

5.1.2.7. EXPECTSTR()

```
std::ostream& gtest_lite::EXPECTSTR (
            const char * exp,
            const char * act,
            bool(*)(const char *, const char *) pred,
            const char * file,
            int line,
            const char * expr,
            const char * lhs = "elvart",
            const char * rhs = "aktual" )  [inline]
```

stringek összehasonlításához. azért nem spec. mert a sima EQ-ra másként kell működnie.

5.1.2.8. ge()

```
template<typename T1 , typename T2 >
bool gtest_lite::ge (
            T1 a,
            T2 b )
```

5.1.2.9. gt()

```
template<typename T1 , typename T2 >
bool gtest_lite::gt (
            T1 a,
            T2 b )
```

5.1.2.10. le()

```
template<typename T1 , typename T2 >
bool gtest_lite::le (
            T1 a,
            T2 b )
```

5.1.2.11. lt()

```
template<typename T1 , typename T2 >
bool gtest_lite::lt (
            T1 a,
            T2 b )
```

5.1.2.12. ne()

```
template<typename T1 , typename T2 >
bool gtest_lite::ne (
            T1 a,
            T2 b )
```

5.1.2.13. nestr()

```
bool gtest_lite::nestr (
            const char * a,
            const char * b )  [inline]
```

6. fejezet

Osztályok dokumentációja

6.1. _Is_Types< F, T > struktúrasablon-referencia

Segédsablon típuskonverzió futás közbeni ellenőrzésere.

```
#include <gtest_lite.h>
```

Statikus publikus tagfüggvények

- template<typename D >
  static char(& f (D))[1]
- template<typename D >
  static char(& f (...))[2]

Statikus publikus attribútumok

- static bool const convertable = sizeof(f<T>(F())) == 1

6.1.1. Részletes leírás

template<typename F, typename T>
struct _Is_Types< F, T >

Segédsablon típuskonverzió futás közbeni ellenőrzésere.

6.1.2. Tagfüggvények dokumentációja

6.1.2.1. f() [1/2]

```
template<typename F , typename T >
template<typename D >
static char(& _Is_Types< F, T >::f (
              ... ))[2]  [static]
```

6.1.2.2. f() [2/2]

```
template<typename F , typename T >
template<typename D >
static char(& _Is_Types< F, T >::f (
             D  ))[1]  [static]
```

6.1.3. Adattagok dokumentációja

6.1.3.1. convertable

```
template<typename F , typename T >
bool const _Is_Types< F, T >::convertable = sizeof(f<T>(F())) == 1  [static]
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- gtest_lite.h

6.2. Jatek osztályreferencia

```
#include <jatek.h>
```

A Jatek osztály együttműködési diagramja:

6.3. Jatekos osztályreferencia

```
#include <jatekos.hpp>
```

Publikus tagfüggvények

- Jatekos (const String &n, Targy ∗t)
- Jatekos (const Jatekos &j)

  Copy ctor.
- Jatekos ∗ copy ()

  copy fv, napló feltöltéshez
- void setTargy (Targy ∗t)
- String getNev () const
- Stat getStat () const
- void setStat (const Stat &s)
- void setStat (const String &str)
- Targy ∗ getTargy () const
- void frissit (char T, bool nyert)
- bool operator== (const Jatekos &j) const
- ∼Jatekos ()

  Destruktor.

6.3.1. Részletes leírás

A Játékos osztály. Egy játékos adatait tárolja (név, használt tárgy).

6.3.2. Konstruktorok és destruktorok dokumentációja

6.3.2.1. Jatekos() [1/2]

```
Jatekos::Jatekos (
            const String & n,
            Targy * t )  [inline]
```

Konstruktor

Paraméterek

nev	- A játékos neve
∗*t*	- a tárgy, amit használ

6.3.2.2. Jatekos() [2/2]

```
Jatekos::Jatekos (
            const Jatekos & j )  [inline]
```

Copy ctor.

6.3.2.3. ~Jatekos()

```
Jatekos::~Jatekos ( )  [inline]
```

Destruktor.

6.3.3. Tagfüggvények dokumentációja

6.3.3.1. copy()

```
Jatekos* Jatekos::copy ( )  [inline]
```

copy fv, napló feltöltéshez

6.3.3.2. frissit()

```
void Jatekos::frissit (
            char T,
            bool nyert )  [inline]
```

frissit - Frissíti a játékos statisztikáját

Paraméterek

T	- A tárgyának betűje
nyert	- Megnyerte-e a játékos a mérkőzést

6.3.3.3. getNev()

```
String Jatekos::getNev ( ) const  [inline]
```

getNev - Visszaadja a játékos nevét

6.3.3.4. getStat()

```
Stat Jatekos::getStat ( ) const  [inline]
```

getStat - visszaadja a játékos statisztikáját

6.3.3.5. getTargy()

```
Targy* Jatekos::getTargy ( ) const  [inline]
```

getTargy - visszaadja a játékos tárgyát

6.3.3.6. operator==()

```
bool Jatekos::operator== (
            const Jatekos & j ) const  [inline]
```

A játékosok összehasonlításához == operator overload

Paraméterek

j	A másik játékos

Visszatérési érték

igaz, ha megegyeznek, egyébként hamis

6.3.3.7. setStat() [1/2]

```
void Jatekos::setStat (
            const Stat & s )  [inline]
```

setStat - Beállítja a játékos statisztikáját egy meglévő statisztika alapján

Paraméterek

s	- A statisztika, amit beállítunk

6.3.3.8. setStat() [2/2]

```
void Jatekos::setStat (
            const String & str )  [inline]
```

setStat - Beállítja a játékos statisztikáját egy String alapján

Paraméterek

str	- A taktika String

6.3.3.9. setTargy()

```
void Jatekos::setTargy (
            Targy * t )  [inline]
```

setItem - beállítja a (már létező) játékos tárgyát

Paraméterek

*t	- a tárgy, amit beállítunk neki

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- jatekos.hpp

6.4. Ko osztályreferencia

```
#include <ko.hpp>
```

A Ko osztály származási diagramja:

A Ko osztály együttműködési diagramja:

Publikus tagfüggvények

- Ko ()
- Ko * copy ()
  copy függvény
- targyak getSelf () const
- targyak getCounter () const

6.4.1. Részletes leírás

A Kő osztály. A papír legyőzi, viszont az ollót ez győzi le.

6.4.2. Konstruktorok és destruktorok dokumentációja

6.4.2.1. Ko()

```
Ko::Ko ( )  [inline]
```

Konstruktor Eltérő nevet meg tárgy típusokat állít be, mint a Tárgy osztály más utódai

6.4.3. Tagfüggvények dokumentációja

6.4.3.1. copy()

```
Ko* Ko::copy ( )  [inline], [virtual]
```

copy függvény

Megvalósítja a következőket: Targy.

6.4.3.2. getCounter()

```
targyak Ko::getCounter ( ) const  [inline], [virtual]
```

getCounter - visszaadja azt a tárgyat, ami legyőzi

Megvalósítja a következőket: Targy.

6.4.3.3. getSelf()

```
targyak Ko::getSelf ( ) const  [inline], [virtual]
```

getTargy - visszaadja a tárgyat

Megvalósítja a következőket: Targy.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- ko.hpp

6.5. Naplo osztályreferencia

```
#include <naplo.h>
```

Publikus tagfüggvények

- Naplo (size_t siz=0)
- Naplo (const Naplo &naplo)

  Copy ctor.
- Naplo & operator= (const Naplo &naplo)

  =operátor
- bool load ()
- bool save ()
- size_t getSize () const
- size_t index (const Jatekos &j)
- Jatekos * operator[ ] (size_t i)

  [] operátor indexeléshez
- Jatekos *const operator[ ] (size_t i) const

  konstans [] operátor indexeléshez
- void hozzaad (Jatekos *j)
- void frissit (Jatekos *j, bool nyert)
- void urites ()
- void topkiir ()
- void sort ()
- ~Naplo ()

  Destruktor.

6.5.1. Részletes leírás

Naplo osztály - A játékosok statisztikáit tároljuk benne

6.5.2. Konstruktorok és destruktorok dokumentációja

6.5.2.1. Naplo() [1/2]

```
Naplo::Naplo (
            size_t siz = 0 )  [inline]
```

Konstruktor

Paraméterek

size	- Megadható a fix tömb mérete.

< Dinamikusan lefoglaljuk a kívánt méretű helyet

6.5.2.2. Naplo() [2/2]

```
Naplo::Naplo (
            const Naplo & naplo )
```

Copy ctor.

6.5.2.3. ~Naplo()

```
Naplo::~Naplo ( )  [inline]
```

Destruktor.

6.5.3. Tagfüggvények dokumentációja

6.5.3.1. frissit()

```
void Naplo::frissit (
            Jatekos * j,
            bool nyert )
```

frissit - Frissíti a napló tartalmát aszerint, hogy nyert-e az adott játékos vagy sem.

Paraméterek

j	- A frissítendő játékos
nyert	- logikai változó, azt tárolja, hogy nyert-e a játékos vagy vesztett.

< Ha nincs benne a játékos, akkor hozzáveszi

< Frissíti a statisztikát

6.5.3.2. getSize()

```
size_t Naplo::getSize ( ) const  [inline]
```

getSize - a naplóban tárolt elemeket adja vissza

Visszatérési érték

n - Ahány ember benne van a naplóban

6.5.3.3. hozzaad()

```
void Naplo::hozzaad (
            Jatekos * j )
```

hozzaad - hozzáad egy új játékost a naplóhoz. Először megnézi, hogy benne van-e, ha nincs, hozzáadja, ha igen, akkor nem csinál semmit

Paraméterek

j	A betenni kívánt játékos

< Ha megtelik a napló...

< 10-zel megnöveljük a napló méretét

< Átmásoljuk az elemeket

< A régi tömböt töröljük

< Átállítjuk a pointert

< Ha nincs benne a játékos és nincs tele a napló, akkor csak hozzáteszi

6.5.3.4. index()

```
size_t Naplo::index (
            const Jatekos & j )
```

index - egy játékos helyét adja meg a tömbben Ha nincs benne, -1-gyel tér vissza

Paraméterek

j	- A keresett játékos

<Ha megegyezik a két játékos, akkor visszatér az aktuális index-szel

6.5.3.5. load()

```
bool Naplo::load ( )
```

load - Betölti a naplóban tárolt játékosokat a naplo.txt fájlból Ha nem létezik, üres lesz a Napló

Visszatérési érték

A sikerességtől függően igaz / hamis

<Ha nem létezik a fájl, akkor kivételt dob

<Az első sor a játékosok számát tartalmazza

< Üres fájl esetén nincs tömbváltoztatás

< Kiürítjük a tömböt

< Töröljük a tömböt

< Új méretet foglalunk

< Beolvassuk a nevét

< Beolvassuk a taktikáját

< A napló megfelelő helyére betesszük a játékos pointerét

< Beállítjuk a taktikáját

6.5.3.6. operator=()

```
Naplo & Naplo::operator= (
            const Naplo & naplo )
```

=operátor

6.5.3.7. operator[]() [1/2]

```
Jatekos* Naplo::operator[] (
            size_t i )  [inline]
```

[] operátor indexeléshez

6.5.3.8. operator[]() [2/2]

```
Jatekos* const Naplo::operator[] (
            size_t i ) const  [inline]
```

konstans [] operátor indexeléshez

6.5.3.9. save()

```
bool Naplo::save ( )
```

save - Elmenti a napló adatait a naplo.txt fájlba (felülírja) Ha nem létezik a txt fájl, generál egyet.

Visszatérési érték

A sikerességtől függően igaz / hamis

< Rendezzük a tömböt növekvő sorrendbe

< Először a játékosok számát írjuk bele

< Nev Taktika formátumban írjuk ki

6.5.3.10. sort()

```
void Naplo::sort ( )
```

sort - rendezi a tömböt növekvő sorrendbe A top 10 kiírása előtt mindig rendez. A selection sort elvén alapszik. < Üres tömb esetén nincs mit csinálni

6.5.3.11. topkiir()

```
void Naplo::topkiir ( )
```

topkiir - Kiírja a 10 legjobb játékos statisztikáját.

topkiir - Kiírja a 10 legjobb játékos statisztikáját növekvő sorrendben. Először rendezi a listát növekvő sorrendbe a sort() függvénnyel.

Paraméterek

os	- A stream, amire ki akarunk írni

< Ha nincs játékos a naplóban, akkor nincs értelme sort-olni vagy kiírni

< Rendezés

< Ha nincs legalább 10 játékos, akkor csak az n méretéig megyünk el

6.5.3.12. urites()

```
void Naplo::urites ( )
```

urites - Törli a napló teljes tartalmát.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- naplo.h
- naplo.cpp

6.6. Ollo osztályreferencia

```
#include <ollo.hpp>
```

Az Ollo osztály származási diagramja:

Az Ollo osztály együttműködési diagramja:

Publikus tagfüggvények

- Ollo ()
- Ollo * copy ()

  copy függvény
- targyak getSelf () const
- targyak getCounter () const

6.6.1. Részletes leírás

Az Olló osztály. A kő legyőzi, viszont a papírt ez győzi le.

6.6.2. Konstruktorok és destruktorok dokumentációja

6.6.2.1. Ollo()

```
Ollo::Ollo ( )  [inline]
```

Konstruktor Eltérő nevet meg tárgy típusokat állít be, mint a Tárgy osztály más utódai

6.6.3. Tagfüggvények dokumentációja

6.6.3.1. copy()

```
Ollo* Ollo::copy ( )  [inline], [virtual]
```

copy függvény

Megvalósítja a következőket: Targy.

6.6.3.2. getCounter()

```
targyak Ollo::getCounter ( ) const  [inline], [virtual]
```

getCounter - visszaadja azt a tárgyat, ami legyőzi

Megvalósítja a következőket: Targy.

6.6.3.3. getSelf()

```
targyak Ollo::getSelf ( ) const  [inline], [virtual]
```

getTargy - visszaadja a tárgyat

Megvalósítja a következőket: Targy.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- ollo.hpp

6.7. Papir osztályreferencia

```
#include <papir.hpp>
```

A Papir osztály származási diagramja:

A Papir osztály együttműködési diagramja:

Publikus tagfüggvények

- Papir ()
- Papir * copy ()

  copy függvény
- targyak getSelf () const
- targyak getCounter () const

6.7.1. Részletes leírás

A Papír osztály. Az olló legyőzi, viszont az követ ez győzi le.

6.7.2. Konstruktorok és destruktorok dokumentációja

6.7.2.1. Papir()

```
Papir::Papir ( )  [inline]
```

Konstruktor Eltérő nevet meg tárgy típusokat állít be, mint a Tárgy osztály más utódai

6.7.3. Tagfüggvények dokumentációja

6.7.3.1. copy()

```
Papir* Papir::copy ( )  [inline], [virtual]
```

copy függvény

Megvalósítja a következőket: Targy.

6.7.3.2. getCounter()

```
targyak Papir::getCounter ( ) const  [inline], [virtual]
```

getCounter - visszaadja azt a tárgyat, ami legyőzi

Megvalósítja a következőket: Targy.

6.7.3.3. getSelf()

```
targyak Papir::getSelf ( ) const  [inline], [virtual]
```

getTargy - visszaadja a tárgyat

Megvalósítja a következőket: Targy.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- papir.hpp

6.8. Stat osztályreferencia

< operator kiíráshoz

```
#include <stat.h>
```

Publikus tagfüggvények

- Stat ()
- int getGyozelmek () const
- String getTaktika () const
- void setStat (const Stat &s)
- void setStat (const String &str)
- void frissit (char T, bool nyert)

6.8.1. Részletes leírás

< operator kiíráshoz

A Statisztika osztály. Egy játékos adatait tartalmazza a mérkőzéseket illetően.

6.8.2. Konstruktorok és destruktorok dokumentációja

6.8.2.1. Stat()

```
Stat::Stat ( )  [inline]
```

Konstruktor

6.8.3. Tagfüggvények dokumentációja

6.8.3.1. frissit()

```
void Stat::frissit (
            char T,
            bool nyert )
```

frissit - Frissíti a játékos statisztikáját

Paraméterek

T	- milyen tárgyat használt
nyert	- nyert-e a játékos

6.8.3.2. getGyozelmek()

```
int Stat::getGyozelmek ( ) const  [inline]
```

6.8.3.3. getTaktika()

```
String Stat::getTaktika ( ) const
```

getTaktika - visszaadja a játékos taktikáját < Ha nincs elmentett taktikája, akkor nem '-' jelet adunk vissza

6.8.3.4. setStat() [1/2]

```
void Stat::setStat (
            const Stat & s )
```

setStat - átállítja a statisztikát egy meglévő statisztika alapján

Paraméterek

s	- A beállítandó statisztika

6.8.3.5. setStat() [2/2]

```
void Stat::setStat (
            const String & str )
```

setStat - átállítja a statisztikát egy String alapján

Paraméterek

str	- A taktika String

<Ha nincs tárolt taktikája, akkor 0 értéket állít be

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- stat.h
- stat.cpp

6.9. String osztályreferencia

`#include <string5.h>`

Publikus tagfüggvények

- String ()
  Paraméter nélküli konstruktor:
- size_t size () const
- const char * c_str () const
- String (char c)
  Ctor.
- String (const char *str)
  Ctor.
- String (const String &str)
  Copy.
- String & operator= (const String &str)
  operator=
- String operator+ (char c) const
- String operator+ (const String &str) const
- const char & operator[ ] (size_t i) const
  [] operator
- char & operator[ ] (size_t i)
  [] operator
- bool operator== (const String &str) const
  == operator
- ~String ()
  Dtor.

6.9.1. Részletes leírás

A String osztály. A 'pData'-ban vannak a karakterek (a lezáró nullával együtt), 'len' a hossza. A hosszba nem számít bele a lezáró nulla.

6.9.2. Konstruktorok és destruktorok dokumentációja

6.9.2.1. String() [1/4]

`String::String ( )  [inline]`

Paraméter nélküli konstruktor:

6.9.2.2. String() [2/4]

```
String::String (
            char c )
```

Ctor.

Konstruktor karakterből.

6.9.2.3. String() [3/4]

```
String::String (
            const char * str )
```

Ctor.

Konstruktor karaktertömbből.

6.9.2.4. String() [4/4]

```
String::String (
            const String & str )
```

Copy.

Másoló konstruktor, String-ből készíti.

6.9.2.5. ~String()

```
String::~String ( )
```

Dtor.

Destruktor.

6.9.3. Tagfüggvények dokumentációja

6.9.3.1. c_str()

```
const char* String::c_str ( ) const  [inline]
```

C-sztringet ad vissza

Visszatérési érték

pointer a tárolt, vagy azzal azonos tartalmú nullával lezárt sztring-re.

6.9.3.2. operator+() [1/2]

```
String String::operator+ (
            char c ) const
```

- operator
- operátor: String-hez jobbról karaktert ad

6.9.3.3. operator+() [2/2]

```
String String::operator+ (
            const String & str ) const
```

- operator
- operátor: String-hez String-et ad (addString)

6.9.3.4. operator=()

```
String & String::operator= (
            const String & str )
```

operator=

= operátor

6.9.3.5. operator==()

```
bool String::operator== (
            const String & str ) const
```

== operator

6.9.3.6. operator[]() [1/2]

```
char & String::operator[] (
            size_t i )
```

[] operator

6.9.3.7. operator[]() [2/2]

```
const char & String::operator[] (
            size_t i ) const
```

[] operator

[] operátorok: egy megadott indexű elem REFERENCIÁJÁVAL térnek vissza indexhiba esetén const char * kivételt dob!

6.9.3.8. size()

```
size_t String::size ( ) const  [inline]
```

Sztring hosszát adja vissza.

Visszatérési érték

sztring tényleges hossza (lezáró nulla nélkül).

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- string5.h
- string5.cpp

6.10. Targy osztályreferencia

```
#include <targy.hpp>
```

A Targy osztály származási diagramja:

Publikus tagfüggvények

- Targy (const String &nev)
- Targy (const Targy &targy)
- virtual targyak getSelf () const =0

  A saját tárgy típusát adja vissza.
- virtual Targy * copy ()=0

  copy függvény
- virtual targyak getCounter () const =0

  A tárgyat legyőző rágy típusát adja vissza.
- char getBetu () const

  A tárgy kezdőbetűjét adja vissza.
- String getNev () const

  A tárgy nevét adja vissza.
- virtual ~Targy ()

6.10.1. Részletes leírás

A Targy ősosztály. Ebből származik a Ko, Papir és Ollo osztály

6.10.2. Konstruktorok és destruktorok dokumentációja

6.10.2.1. Targy() [1/2]

```
Targy::Targy (
            const String & nev )  [inline]
```

Konstruktor

Paraméterek

nev	- a tárgy neve, kiírásokhoz kell majd

6.10.2.2. Targy() [2/2]

```
Targy::Targy (
            const Targy & targy )  [inline]
```

6.10.2.3. ~Targy()

```
virtual Targy::~Targy ( )  [inline], [virtual]
```

6.10.3. Tagfüggvények dokumentációja

6.10.3.1. copy()

```
virtual Targy* Targy::copy ( )  [pure virtual]
```

copy függvény

Megvalósítják a következők: Papir, Ollo és Ko.

6.10.3.2. getBetu()

```
char Targy::getBetu ( ) const  [inline]
```

A tárgy kezdőbetűjét adja vissza.

6.10.3.3. getCounter()

```
virtual targyak Targy::getCounter ( ) const  [pure virtual]
```

A tárgyat legyőző rágy típusát adja vissza.

Megvalósítják a következők: Papir, Ollo és Ko.

6.10.3.4. getNev()

```
String Targy::getNev ( ) const  [inline]
```

A tárgy nevét adja vissza.

6.10.3.5. getSelf()

```
virtual targyak Targy::getSelf ( ) const  [pure virtual]
```

A saját tárgy típusát adja vissza.

Megvalósítják a következők: Papir, Ollo és Ko.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- targy.hpp

6.11. gtest_lite::Test struktúrareferencia

```
#include <gtest_lite.h>
```

Publikus tagfüggvények

- void begin (const char *n)

  Teszt kezdete.
- std::ostream & end (bool memchk=false)

  Teszt vége.
- bool fail ()
- bool astatus ()
- std::ostream & expect (bool st, const char *file, int line, const char *expr, bool pr=false)

  Eredményt adminisztráló tagfüggvény True a jó eset.
- ~Test ()

  Destruktor.

Statikus publikus tagfüggvények

- static Test & getTest ()

Publikus attribútumok

- int sum

  tesztek számlálója
- int failed

  hibás tesztek
- int ablocks

  allokált blokkok száma
- bool status

  éppen futó teszt státusza.
- bool tmp

  temp a kivételkezeléshez;
- std::string name

  éppen futó teszt neve.
- std::fstream null

  nyelő, ha nem kell kiírni semmit

6.11.1. Részletes leírás

Tesztek állapotát tároló osztály. Egyetlen egy statikus példány keletkezik, aminek a destruktora a futás végén hívódik meg.

6.11.2. Konstruktorok és destruktorok dokumentációja

6.11.2.1. ~Test()

```
gtest_lite::Test::~Test ( )  [inline]
```

Destruktor.

6.11.3. Tagfüggvények dokumentációja

6.11.3.1. astatus()

```
bool gtest_lite::Test::astatus ( )  [inline]
```

6.11.3.2. begin()

```
void gtest_lite::Test::begin (
            const char * n )  [inline]
```

Teszt kezdete.

6.11.3.3. end()

```
std::ostream& gtest_lite::Test::end (
            bool memchk = false )  [inline]
```

Teszt vége.

6.11.3.4. expect()

```
std::ostream& gtest_lite::Test::expect (
            bool st,
            const char * file,
            int line,
            const char * expr,
            bool pr = false )  [inline]
```

Eredményt adminisztráló tagfüggvény True a jó eset.

6.11.3.5. fail()

```
bool gtest_lite::Test::fail ( )  [inline]
```

6.11.3.6. getTest()

```
static Test& gtest_lite::Test::getTest ( )  [inline], [static]
```

< egyedüli (singleton) példány

6.11.4. Adattagok dokumentációja

6.11.4.1. ablocks

```
int gtest_lite::Test::ablocks
```

allokált blokkok száma

6.11.4.2. failed

```
int gtest_lite::Test::failed
```

hibás tesztek

6.11.4.3. name

```
std::string gtest_lite::Test::name
```

éppen futó teszt neve.

6.11.4.4. null

```
std::fstream gtest_lite::Test::null
```

nyelő, ha nem kell kiírni semmit

6.11.4.5. status

```
bool gtest_lite::Test::status
```

éppen futó teszt státusza.

6.11.4.6. sum

```
int gtest_lite::Test::sum
```

tesztek számlálója

6.11.4.7. tmp

```
bool gtest_lite::Test::tmp
```

temp a kivételkezeléshez;

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- gtest_lite.h

7. fejezet

Fájlok dokumentációja

7.1. gtest_lite.h fájlreferencia

```
#include <iostream>
#include <cassert>
#include <cmath>
#include <cstring>
#include <limits>
#include <cstdlib>
#include <string>
#include <fstream>
```

A gtest_lite.h definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- struct _Is_Types< F, T >

  Segédsablon típuskonverzió futás közbeni ellenőrzésere.
- struct gtest_lite::Test

Névterek

- gtest_lite

  gtest_lite: a keretrendszer függvényinek és objektumainak névtere

Makródefiníciók

- #define TEST(C, N) do { gtest_lite::test.begin(#C"."#N);
- #define END gtest_lite::test.end(); } while (false);

  Teszteset vége.
- #define ENDM gtest_lite::test.end(true); } while (false);
- #define ENDMsg(t) gtest_lite::test.end(true) << t << std::endl; } while (false);
- #define SUCCEED() gtest_lite::test.expect(true, __FILE__, __LINE__, "SUCCEED()", true)

  Sikeres teszt makrója.
- #define FAIL() gtest_lite::test.expect(false, __FILE__, __LINE__, "FAIL()", true)

Sikertelen teszt fatális hiba makrója.

- #define ADD_FAILURE() gtest_lite::test.expect(false, __FILE__, __LINE__, "ADD_FAILURE()", true)

  Sikertelen teszt makrója.

- #define EXPECT_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::eq, __FILE__, _↩ _LINE__, "EXPECT_EQ(" #expected ", " #actual ")")

  Azonosságot elváró makró

- #define EXPECT_NE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::ne, __FILE__, _↩ _LINE__, "EXPECT_NE(" #expected ", " #actual ")", "etalon")

  Eltérést elváró makró

- #define EXPECT_LE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::le, __FILE__, __↩ LINE__, "EXPECT_LE(" #expected ", " #actual ")", "etalon")

  Kisebb, vagy egyenlő relációt elváró makró

- #define EXPECT_LT(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::lt, __FILE__, __↩ LINE__, "EXPECT_LT(" #expected ", " #actual ")", "etalon")

  Kisebb, mint relációt elváró makró

- #define EXPECT_GE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::ge, __FILE__, _↩ _LINE__, "EXPECT_GE(" #expected ", " #actual ")", "etalon")

  Nagyobb, vagy egyenlő relációt elváró makró

- #define EXPECT_GT(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::gt, __FILE__, __↩ LINE__, "EXPECT_GT(" #expected ", " #actual ")", "etalon")

  Nagyobb, mint relációt elváró makró

- #define EXPECT_TRUE(actual) gtest_lite::EXPECT_(true, actual, gtest_lite::eq, __FILE__, __LINE__↩ , "EXPECT_TRUE(" #actual ")")

  Igaz értéket elváró makró

- #define EXPECT_FALSE(actual) gtest_lite::EXPECT_(false, actual, gtest_lite::eq, __FILE__, __LINE__↩ , "EXPECT_FALSE(" #actual ")")

  Hamis értéket elváró makró

- #define EXPECT_FLOAT_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, __LINE__, "EXPECT_FLOAT_EQ(" #expected ", " #actual ")")

  Valós számok azonosságát elváró makró

- #define EXPECT_DOUBLE_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, __LINE__, "EXPECT_DOUBLE_EQ(" #expected ", " #actual ")")

  Valós számok azonosságát elváró makró

- #define EXPECT_STREQ(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstr, __↩ FILE__, __LINE__, "EXPECT_STREQ(" #expected ", " #actual ")")

  C stringek (const char ∗) azonosságát tesztelő makró

- #define EXPECT_STRNE(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestr, __↩ FILE__, __LINE__, "EXPECT_STRNE(" #expected ", " #actual ")", "etalon")

  C stringek (const char ∗) eltéréset tesztelő makró

- #define EXPECT_STRCASEEQ(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_STRCASEEQ(" #expected ", " #actual ")")

  C stringek (const char ∗) azonosságát tesztelő makró (kisbetű/nagybetű azonos)

- #define EXPECT_STRCASENE(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite↩ ::nestrcase, __FILE__, __LINE__, "EXPECT_STRCASENE(" #expected ", " #actual ")", "etalon")

  C stringek (const char ∗) eltéréset tesztelő makró (kisbetű/nagybetű azonos)

- #define EXPECT_THROW(statement, exception_type)

  Kivételt várunk.

- #define EXPECT_ANY_THROW(statement)

  Kivételt várunk.

- #define EXPECT_NO_THROW(statement)

  Nem várunk kivételt.

- #define ASSERT_NO_THROW(statement)

Nem várunk kivételt.

- #define EXPECT_THROW_THROW(statement, exception_type)

  Kivételt várunk és továbbdobjuk – ilyen nincs a gtest-ben.
- #define EXPECT_ENVEQ(expected, actual) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstr, __FILE__, __LINE__, "EXPECT_ENVEQ(" #expected ", " #actual ")")

  Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben.
- #define EXPECT_ENVCASEEQ(expected, actual) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_ENVCASEEQ(" #expected ", " #actual ")")

  Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben (kisbetű/nagybetű azonos)
- #define ASSERT_EQ(expected, actual) gtest_lite::ASSERT_(expected, actual, gtest_lite::eq, "ASSER_EQ")

  Azonosságot elváró makró
- #define ASSERT_NO_THROW(statement)

  Nem várunk kivételt.
- #define CREATE_Has_(X)
- #define EXPECTTHROW(statement, exp, act)

  EXPECTTHROW: kivételkezelés.
- #define ASSERTTHROW(statement, exp, act)
- #define ASSERT_(expected, actual, fn, op)
- #define GTINIT(IS)
- #define GTEND(os)

Függvények

- void hasMember (...)
- template<typename T1 , typename T2 >
  std::ostream & gtest_lite::EXPECT_ (T1 exp, T2 act, bool(∗pred)(T1, T2), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")

  általános sablon a várt értékhez.
- template<typename T1 , typename T2 >
  std::ostream & gtest_lite::EXPECT_ (T1 ∗exp, T2 ∗act, bool(∗pred)(T1 ∗, T2 ∗), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")

  pointerre specializált sablon a várt értékhez.
- std::ostream & gtest_lite::EXPECTSTR (const char ∗exp, const char ∗act, bool(∗pred)(const char ∗, const char ∗), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")
- template<typename T1 , typename T2 >
  bool gtest_lite::eq (T1 a, T2 b)
- bool gtest_lite::eqstr (const char ∗a, const char ∗b)
- bool gtest_lite::eqstrcase (const char ∗a, const char ∗b)
- template<typename T1 , typename T2 >
  bool gtest_lite::ne (T1 a, T2 b)
- bool gtest_lite::nestr (const char ∗a, const char ∗b)
- template<typename T1 , typename T2 >
  bool gtest_lite::le (T1 a, T2 b)
- template<typename T1 , typename T2 >
  bool gtest_lite::lt (T1 a, T2 b)
- template<typename T1 , typename T2 >
  bool gtest_lite::ge (T1 a, T2 b)
- template<typename T1 , typename T2 >
  bool gtest_lite::gt (T1 a, T2 b)
- template<typename T >
  bool gtest_lite::almostEQ (T a, T b)

7.1.1. Részletes leírás

(v3/2019)

Google gtest keretrendszerhez hasonló rendszer. Sz.I. 2015., 2016., 2017. (_Has_X) Sz.I. 2018 (template), ENDM, ENDMsg, nullptr_t Sz.I. 2019 singleton Sz.I. 2021 ASSERT.., STRCASE...

A tesztelés legalapvetőbb funkcióit támogató függvények és makrók. Nem szálbiztos megvalósítás.

Szabadon felhasználható, bővíthető.

Használati példa: Teszteljük az f(x)=2∗x függvényt: int f(int x) { return 2∗x; }

int main() { TEST(TeszEsetNeve, TesztNeve) EXPECT_EQ(0, f(0)); EXPECT_EQ(4, f(2)) << "A függvény hibás eredményt adott" << std::endl; ... END ... // Fatális hiba esetén a teszteset nem fut tovább. Ezek az ASSERT... makrók. // Nem lehet a kiírásukhoz további üzenetet fűzni. PL: TEST(TeszEsetNeve, TesztNeve) ASSERT_NO_THROW(f(0)); // itt nem lehet << "duma" EXPECT_EQ(4, f(2)) << "A függvény hibás eredményt adott" << std::endl; ... END ...

A működés részleteinek megértése szorgalmi feladat.

7.1.2. Makródefiníciók dokumentációja

7.1.2.1. ADD_FAILURE

```
#define ADD_FAILURE( ) gtest_lite::test.expect(false, __FILE__, __LINE__, "ADD_FAILURE()",
true)
```

Sikertelen teszt makrója.

7.1.2.2. ASSERT_

```
#define ASSERT_(
                expected,
                actual,
                fn,
                op )
```

Érték:

```
    EXPECT_(expected, actual, fn, __FILE__, __LINE__, #op "(" #expected ", " #actual ")" ); \
    if (!gtest_lite::test.status) { gtest_lite::test.end(); break; }
```

7.1.2.3. ASSERT_EQ

```
#define ASSERT_EQ(
                expected,
                actual ) gtest_lite::ASSERT_(expected, actual, gtest_lite::eq, "ASSER_EQ")
```

Azonosságot elváró makró

ASSERT típusú ellenőrzések. CSak 1-2 van megvalósítva. Nem ostream& -val térnek vissza !!! Kivételt várunk

7.1.2.4. ASSERT_NO_THROW [1/2]

```
#define ASSERT_NO_THROW(
                statement )
```

Érték:

```
    try { gtest_lite::test.tmp = true; statement; } \
    catch (...) { gtest_lite::test.tmp = false; }\
    ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.1.2.5. ASSERT_NO_THROW [2/2]

```
#define ASSERT_NO_THROW(
                statement )
```

Érték:

```
    try { gtest_lite::test.tmp = true; statement; } \
    catch (...) { gtest_lite::test.tmp = false; }\
    ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.1.2.6. ASSERTTHROW

```
#define ASSERTTHROW(
                statement,
                exp,
                act )
```

Érték:

```
    gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__, __LINE__, #statement) \
    « "** Az utasitas " « (act) \
    « "\n** Azt vartuk, hogy " « (exp) « std::endl; if (!gtest_lite::test.status) { gtest_lite::test.end();
       break; }
```

7.1.2.7. CREATE_Has_

```
#define CREATE_Has_(
                X )
```

Érték:

```
template<typename T> struct _Has_##X {  \
    struct Fallback { int X; };         \
    struct Derived : T, Fallback {};    \
    template<typename C, C> struct ChT; \
    template<typename D> static char (&f(ChT<int Fallback::*, &D::X>*))[1]; \
    template<typename D> static char (&f(...))[2]; \
    static bool const member = sizeof(f<Derived>(0)) == 2; \
};
```

Segédmakró egy adattag, vagy tagfüggvény létezésének tesztelésére futási időben Ötlet: https://cpptalk.wordpress.com/2009/09/12/substitution-failure-is-not-an-error-2

Használat: CREATE_Has_(size) ... if (Has_size<std::string>::member)...

7.1.2.8. END

```
#define END gtest_lite::test.end(); } while (false);
```

Teszteset vége.

7.1.2.9. ENDM

```
#define ENDM gtest_lite::test.end(true); } while (false);
```

Teszteset vége allokált blokkok számának összehasonlításával Ez az ellenőrzés nem bomba biztos.

7.1.2.10. ENDMsg

```
#define ENDMsg(
              t ) gtest_lite::test.end(true) << t << std::endl; } while (false);
```

Teszteset vége allokált blokkok számának összehasonlításával Ez az ellenőrzés nem bomba biztos. Ha hiba van kiírja az üzenetet.

7.1.2.11. EXPECT_ANY_THROW

```
#define EXPECT_ANY_THROW(
              statement )
```

Érték:

```
    try { gtest_lite::test.tmp = false; statement; } \
    catch (...) { gtest_lite::test.tmp = true; } \
    EXPECTTHROW(statement, "kivetelt dob.", "nem dobott kivetelt.")
```

Kivételt várunk.

7.1.2.12. EXPECT_DOUBLE_EQ

```
#define EXPECT_DOUBLE_EQ(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, ↩
__LINE__, "EXPECT_DOUBLE_EQ(" #expected ", " #actual ")" )
```

Valós számok azonosságát elváró makró

7.1.2.13. EXPECT_ENVCASEEQ

```
#define EXPECT_ENVCASEEQ(
              expected,
              actual ) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstrcase,
__FILE__, __LINE__, "EXPECT_ENVCASEEQ(" #expected ", " #actual ")" )
```

Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben (kisbetű/nagybetű azonos)

7.1.2.14. EXPECT_ENVEQ

```
#define EXPECT_ENVEQ(
              expected,
              actual ) gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstr,
__FILE__, __LINE__, "EXPECT_ENVEQ(" #expected ", " #actual ")" )
```

Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben.

7.1.2.15. EXPECT_EQ

```
#define EXPECT_EQ(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::eq, __FILE__, __LINE↩
__, "EXPECT_EQ(" #expected ", " #actual ")" )
```

Azonosságot elváró makró

7.1.2.16. EXPECT_FALSE

```
#define EXPECT_FALSE(
              actual ) gtest_lite::EXPECT_(false, actual, gtest_lite::eq, __FILE__, __LINE__,
"EXPECT_FALSE(" #actual ")" )
```

Hamis értéket elváró makró

7.1.2.17. EXPECT_FLOAT_EQ

```
#define EXPECT_FLOAT_EQ(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, ↩
__LINE__, "EXPECT_FLOAT_EQ(" #expected ", " #actual ")" )
```

Valós számok azonosságát elváró makró

7.1.2.18. EXPECT_GE

```
#define EXPECT_GE(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::ge, __FILE__, __LINE↩
__, "EXPECT_GE(" #expected ", " #actual ")", "etalon" )
```

Nagyobb, vagy egyenlő relációt elváró makró

7.1.2.19. EXPECT_GT

```
#define EXPECT_GT(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::gt, __FILE__, __LINE↩
__, "EXPECT_GT(" #expected ", " #actual ")", "etalon" )
```

Nagyobb, mint relációt elváró makró

7.1.2.20. EXPECT_LE

```
#define EXPECT_LE(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::le, __FILE__, __LINE↩
__, "EXPECT_LE(" #expected ", " #actual ")", "etalon" )
```

Kisebb, vagy egyenlő relációt elváró makró

7.1.2.21. EXPECT_LT

```
#define EXPECT_LT(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::lt, __FILE__, __LINE↩
__, "EXPECT_LT(" #expected ", " #actual ")", "etalon" )
```

Kisebb, mint relációt elváró makró

7.1.2.22. EXPECT_NE

```
#define EXPECT_NE(
              expected,
              actual ) gtest_lite::EXPECT_(expected, actual, gtest_lite::ne, __FILE__, __LINE↩
__, "EXPECT_NE(" #expected ", " #actual ")", "etalon" )
```

Eltérést elváró makró

7.1.2.23. EXPECT_NO_THROW

```
#define EXPECT_NO_THROW(
                statement )
```

Érték:

```
    try { gtest_lite::test.tmp = true; statement; } \
    catch (...) { gtest_lite::test.tmp = false; }\
    EXPECTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.1.2.24. EXPECT_STRCASEEQ

```
#define EXPECT_STRCASEEQ(
                expected,
                actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstrcase, __FILE_↩
_, __LINE__, "EXPECT_STRCASEEQ(" #expected ", " #actual ")" )
```

C stringek (const char ∗) azonosságát tesztelő makró (kisbetű/nagybetű azonos)

7.1.2.25. EXPECT_STRCASENE

```
#define EXPECT_STRCASENE(
                expected,
                actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestrcase, __FILE_↩
_, __LINE__, "EXPECT_STRCASENE(" #expected ", " #actual ")", "etalon" )
```

C stringek (const char ∗) eltéréset tesztelő makró (kisbetű/nagybetű azonos)

7.1.2.26. EXPECT_STREQ

```
#define EXPECT_STREQ(
                expected,
                actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstr, __FILE__, _↩
_LINE__, "EXPECT_STREQ(" #expected ", " #actual ")" )
```

C stringek (const char ∗) azonosságát tesztelő makró

7.1.2.27. EXPECT_STRNE

```
#define EXPECT_STRNE(
                expected,
                actual ) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestr, __FILE__, _↩
_LINE__, "EXPECT_STRNE(" #expected ", " #actual ")", "etalon" )
```

C stringek (const char ∗) eltéréset tesztelő makró

7.1.2.28. EXPECT_THROW

```
#define EXPECT_THROW(
                statement,
                exception_type )
```

Érték:

```
    try { gtest_lite::test.tmp = false; statement; } \
    catch (exception_type) { gtest_lite::test.tmp = true; } \
    catch (...) { } \
    EXPECTTHROW(statement, "kivetelt dob.", "nem dobott '"#exception_type"' kivetelt.")
```

Kivételt várunk.

7.1.2.29. EXPECT_THROW_THROW

```
#define EXPECT_THROW_THROW(
              statement,
              exception_type )
```

Érték:

```
    try { gtest_lite::test.tmp = false; statement; } \
    catch (exception_type) { gtest_lite::test.tmp = true; throw; } \
    EXPECTTHROW(statement, "kivetelt dob.", "nem dobott '"#exception_type"' kivetelt.")
```

Kivételt várunk és továbbdobjuk – ilyen nincs a gtest-ben.

7.1.2.30. EXPECT_TRUE

```
#define EXPECT_TRUE(
              actual ) gtest_lite::EXPECT_(true, actual, gtest_lite::eq, __FILE__, __LINE__↩
, "EXPECT_TRUE(" #actual ")" )
```

Igaz értéket elváró makró

7.1.2.31. EXPECTTHROW

```
#define EXPECTTHROW(
              statement,
              exp,
              act )
```

Érték:

```
    gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__, __LINE__, #statement) \
    « "** Az utasitas " « (act) \
    « "\n** Azt vartuk, hogy " « (exp) « std::endl
```

EXPECTTHROW; kivételkezelés.

Belső megvalósításhoz tartozó makrók, és osztályok.

7.1.2.32. Nem célszerű közvetlenül használni, vagy módosítani

7.1.2.33. FAIL

```
#define FAIL( ) gtest_lite::test.expect(false, __FILE__, __LINE__, "FAIL()", true)
```

Sikertelen teszt fatális hiba makrója.

7.1.2.34. GTEND

```
#define GTEND(
              os )
```

7.1.2.35. GTINIT

```
#define GTINIT(
              IS )
```

7.1.2.36. SUCCEED

```
#define SUCCEED( ) gtest_lite::test.expect(true, __FILE__, __LINE__, "SUCCEED()", true)
```

Sikeres teszt makrója.

7.1.2.37. TEST

```
#define TEST(
             C,
             N ) do { gtest_lite::test.begin(#C"."#N);
```

Teszt kezdete. A makró paraméterezése hasonlít a gtest paraméterezéséhez. Így az itt elkészített testek könnyen átemelhetők a gtest keretrendszerbe.

Paraméterek

C	- teszteset neve (csak a gtest kompatibilitás miatt van külön neve az eseteknek)
N	- teszt neve

7.1.3. Függvények dokumentációja

7.1.3.1. hasMember()

```
void hasMember (
             ... ) [inline]
```

Segédfüggvény egy publikus adattag, vagy tagfüggvény létezésének tesztelésére fordítási időben

7.2. jatek.cpp fájlreferencia

```
#include "jatek.h"
```

A jatek.cpp definíciós fájl függési gráfja:

Függvények

- void clearConsole ()

7.2.1. Részletes leírás

Itt valósulnak meg a mérkõzések demonstrálására használt jatek osztály tagfüggvényei. Ebben a fájlban vannak a függvények megvalósításai.

7.2.2. Függvények dokumentációja

7.2.2.1. clearConsole()

```
void clearConsole ( )
```

clearConsole - A menürendszer szépsége érdekében lehetőség van tisztítani a konzolt

7.3. jatek.h fájlreferencia

```
#include "targy.hpp"
#include "ko.hpp"
#include "papir.hpp"
#include "ollo.hpp"
#include "naplo.h"
#include "memtrace.h"
#include <fstream>
```

A jatek.h definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Jatek

7.3.1. Részletes leírás

Itt valósul meg a mérkőzések demonstrálására használt jatek osztály. Ebben a fájlban vannak a függvények deklarációi és az inline függvények megvalósításai.

7.4. jatekos.hpp fájlreferencia

```
#include "string5.h"
#include "targy.hpp"
#include "stat.h"
#include "memtrace.h"
```

A jatekos.hpp definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Jatekos

Függvények

- std::ostream & operator<< (std::ostream &os, const Jatekos &j)

7.4.1. Függvények dokumentációja

7.4.1.1. operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Jatekos & j )  [inline]
```

Kiíratáshoz << operátor overload

Paraméterek

os	- A kiíráshoz használt stream
j	- A kiírandó játékos (statisztikája)

7.5. ko.hpp fájlreferencia

```
#include "targy.hpp"
```

A ko.hpp definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Ko

7.6. main.cpp fájlreferencia

```
#include <iostream>
```

```
#include "gtest_lite.h"
#include "jatekos.hpp"
#include "naplo.h"
#include "ko.hpp"
#include "papir.hpp"
#include "ollo.hpp"
#include "memtrace.h"
#include "jatek.h"
#include <time.h>
```

A main.cpp definíciós fájl függési gráfja:

Makródefiníciók

- #define TESTING

Függvények

- void test1 ()
- void test2 ()
- void test3 ()
- int main ()

7.6.1. Részletes leírás

A teszteseteket itt ellenõrizzük le.

7.6.2. Makródefiníciók dokumentációja

7.6.2.1. TESTING

```
#define TESTING
```

7.6.3. Függvények dokumentációja

7.6.3.1. main()

```
int main ( )
```

CodeBlocksnál a konzol hibásan jeleníti meg a betűket, tesztelésre javasolt a Visual Studio használata vagy esetleg a Linuxon való futtatás

7.6.3.2. test1()

```
void test1 ( )
```

test1 - betöltés fájlból Először egy nem létező fájlból olvasna, ekkor kivételt várunk. Ezután a már létező fájból olvasna, ami üres, ekkor azt várjuk, hogy hamissal térjen vissza Ezt követően a mentést próbáljuk meg, ekkor azt várjuk, hogy sikerüljön. Végül < Ezzel a sorral létrejön a fájl, még akkor is, ha ezelőtt nem létezett, viszont üres!

7.6.3.3. test2()

```
void test2 ( )
```

test2 - Játékosok hozzáadása nyilvántartáshoz, (napló bővítése), napló ürítése és a legjobbak kiírása. Először egy nem létező játékost ad hozzá, ekkor azt várjuk, hogy sikerüljön, a program azt írja ki, hogy "Játékos felvéve". Ezután egy már létező játékos ad hozzá, ekkor nem várunk semmit. Ezt követően kiírjuk a top 10 (vagy annál kevesebb) legjobb játékost. Végül ürítjük a naplót, majd ezt követően az ismételt kiírásnál azt várjuk, hogy azt írja ki a program, hogy "Nincs játékos a naplóban" FIGYELEM: Ez a teszteset nem a gtest_lite-val fut. < "Játékos felvéve"

< Nincs kiírás, mert már létező játékos

7.6.3.4. test3()

`void test3 ( )`
test3 - egy mérkőzés levezénylése Először ugyanazt a játékost adjuk meg mindkétszer a játékosok bekérésénél, itt kivételt várunk. Ezután ha nem érkezik kivétel (itt már különböző játékosokat adunk meg!), akkor jól levezényelt egy mérkőzést és jól frissítette a naplót

7.7. memtrace.cpp fájlreferencia

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
```
A memtrace.cpp definíciós fájl függési gráfja:

7.8. memtrace.h fájlreferencia

Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

7.9. naplo.cpp fájlreferencia

```
#include "naplo.h"
#include <iostream>
#include <fstream>
```
A naplo.cpp definíciós fájl függési gráfja:

7.9.1. Részletes leírás

Ebbe a fájlba kerül a statisztikákat tároló napló osztály tagfüggvényeinek megvalósítása.

7.10. naplo.h fájlreferencia

```
#include "jatekos.hpp"
#include "memtrace.h"
```
A naplo.h definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Naplo

Enumerációk

- enum BOOL { DRAW = -1 , TRUE = 1 , FALSE = 0 }

7.10.1. Részletes leírás

Ebbe a fájlba kerül a statisztikákat tároló napló osztály deklarációi, inline tagfüggvényei.

7.10.2. Enumerációk dokumentációja

7.10.2.1. BOOL

```
enum BOOL
```
BOOL - saját logikai típus mérkőzések lezárására. TRUE az igaz, FALSE a hamis és DRAW a döntetlen

Enumeráció-értékek

DRAW	
TRUE	
FALSE	

7.11. ollo.hpp fájlreferencia

```
#include "targy.hpp"
```
Az ollo.hpp definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Ollo

7.12. papir.hpp fájlreferencia

```
#include "targy.hpp"
```
A papir.hpp definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Papir

7.13. stat.cpp fájlreferencia

```
#include "stat.h"
```
A stat.cpp definíciós fájl függési gráfja:

Függvények

- std::ostream & operator<< (std::ostream &os, const Stat &s)

7.13.1. Részletes leírás

Ebbe a fájlba kerül a Statisztika osztály tagfüggvényeinek megvalósítása.

7.13.2. Függvények dokumentációja

7.13.2.1. operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Stat & s )
```

7.14. stat.h fájlreferencia

```
#include "string5.h"
```

A stat.h definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Stat

  < *operator kiíráshoz*

Függvények

- std::ostream & operator<< (std::ostream &os, const Stat &s)

7.14.1. Részletes leírás

Itt valósul meg a statisztikák tárolására alkalmas stat osztály. Ebben a fájlban a függvények deklarációi szereplnek, illetve az inline tagfüggvények.

7.14.2. Függvények dokumentációja

7.14.2.1. operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Stat & s )
```

7.15. string5.cpp fájlreferencia

```
#include <iostream>
#include <cstring>
#include "string5.h"
```

A string5.cpp definíciós fájl függési gráfja:

Függvények

- String operator+ (char c, const String &str)

  char-hoz String-et ad
- std::ostream & operator<< (std::ostream &os, const String &str)

  << *operator, ami kiír az ostream-re*
- std::istream & operator>> (std::istream &is, String &s0)

  operator, ami beolvas az istreamről egy szót

7.15.1. Részletes leírás

A string osztály megvalósítása saját módszerekkel.

7.15.2. Függvények dokumentációja

7.15.2.1. operator+()

```
String operator+ (
            char c,
            const String & str )
```

char-hoz String-et ad
char + str

7.15.2.2. operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const String & str )
```

<< operator, ami kiír az ostream-re

7.15.2.3. operator>>()

```
std::istream& operator>> (
            std::istream & is,
            String & s0 )
```

operator, ami beolvas az istreamről egy szót

operator

7.16. string5.h fájlreferencia

`#include <iostream>`
A string5.h definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class String

Függvények

- String operator+ (char c, const String &str)

  char + str
- std::ostream & operator<< (std::ostream &os, const String &str)

  << operator, ami kiír az ostream-re
- std::istream & operator>> (std::istream &is, String &s0)

  operator

7.16.1. Részletes leírás

Ez a fájl tartalmazza a

- String osztály deklarációját
- az inline függvényeket.

7.16.2. Függvények dokumentációja

7.16.2.1. operator+()

```
String operator+ (
            char c,
            const String & str )
```

char + str
< operator
char + str

7.16.2.2. operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const String & str )
```

<< operator, ami kiír az ostream-re

7.16.2.3. operator>>()

```
std::istream& operator>> (
            std::istream & is,
            String & s0 )
```

operator

operator

7.17. targy.hpp fájlreferencia

`#include "string5.h"`
A targy.hpp definíciós fájl függési gráfja: Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:

Osztályok

- class Targy

Enumerációk

- enum targyak { KO =1 , PAPIR =2 , OLLO =3 }

  Ezeket a tárgyakat lehet használni, bővíthető!

7.17.1. Enumerációk dokumentációja

7.17.1.1. targyak

`enum targyak`
Ezeket a tárgyakat lehet használni, bővíthető!

Enumeráció-értékek

KO	
PAPIR	
OLLO	

Tárgymutató