Catch.hpp ismertető

A Programozás beadandókhoz szükséges lesz a megírt kódhoz tesztkörnyezetet írni egységtesztek formájában. A cél ezzel az, hogy a kódot egy irányított környezetben tudjuk tesztelni, ahol a várt eredményeket szembe tudjuk állítani a program által számítottakkal. Ezeknek a teszteknek a megírásához a **catch.hpp** tesztelő környezetet fogjuk használni. Ezt a https://github.com/philsquared/hash_trie githubról tudjátok letölteni. Az alábbi egy „pongyola” ismertető a használathoz.

Használat

A catch.hpp használatához a projektünkbe definiálni kell, hogy ez a program egy Catch tesztkörnyezet lesz, majd be kell includeolni a catch.hpp fájlt. Ezt a #define CATCH_CONFIG_MAIN paranccsal tehetjük meg, illetve a szokásos include-al.

```
#define CATCH_CONFIG_MAIN
#include "catch.hpp"
#include <iostream>
#include "enor.h"
#include "tetelek.h"

TEST_CASE("Maximum az elejen", "[Maxelol.txt]")
{
    int maximum; std::string first;
    MaxKiv(maximum,first, "Maxelol.txt");
    CHECK(first == "Bela");
}

TEST_CASE("Maximum a kozepen", "[Maxkozepen.txt]")
{
    int maximum; std::string first;
    MaxKiv(maximum,first, "Maxkozepen.txt");
    CHECK(first == "Bela");
}

TEST_CASE("Maximum a végén", "[Maxvegen.txt]")
{
    int maximum; std::string first;
    MaxKiv(maximum, first, "Maxvegen.txt");
    CHECK(first == "Adam");
```

Ezen kívül már csak a teszteseteket kell megírni, a fent látható képen az önálló teszteseteket mindig külön TEST_CASE-ként kell definiálni. Ennek a paraméterlistájában ajánlatos megjegyzéseket fűzni a tesztesethez, hogy ez milyen esetet is fed le.

TEST_CASE kódját szabványos C++ kódként írjuk, itt hozzuk létre a tesztelni kívánt objektumokat úgy, hogy a tesztesetet lefedjék (általában ezt külön inputtal érjük el). Ezeket, hogy a teszteknek legyen is értelme, a CHECK paranccsal tudjuk ellenőrizni. A CHECK paraméterének, egy egyszerű, logikai értéket visszaadó utasításnak kell lennie, az egyszerű alatt az értendő, hogy nem tehetünk a CHECK-be ÉS (&&), VAGY(||) stb. logikai műveleteket. A CHECK a futás során megvizsgálja, hogy a paraméterül kapott kifejezés igaz-e. Ha igen, akkor ezt sikeres tesztesetnek tudja be a teszter. Ha nem, akkor megmondja, hogy az adott teszteseten a program megbukott, és megmutatja (ha tudja), hogy milyen értéke volt a változóknak, amiket vizsgáltunk („kibontja” a kifejezést”).

Hogy elkerüljük azt, hogy két külön projektbe kelljen létrehozni ezt a tesztkörnyezeteket és a főprogramot, ezért bevethetünk egy egyszerű trükköt, amivel a main programunk apró módosításával a teszteseteket futtatjuk a főprogram helyett. Ez két futási „módot” ad a programunknak.

```
// Ezzel lehet a manuális es a unit teszt mód között váltogatni
#define NORMAL_MODE
#ifdef NORMAL_MODE

int main()
{
```

```
#else
#define CATCH_CONFIG_MAIN
#include "catch.hpp"
#include "verem.h"

TEST_CASE("konstruktor, ures verem", "[verem]")
{
    SECTION("konstruktor")
    {
```

A fenti képen látjuk, hogy preprocesszor direktívákkal definiálunk egy NORMAL_MODE-ot, majd megnézzük, hogy ez definiált-e. Ha igen, akkor a főprogram kerül futtatásra. Ha nem, akkor a tesztelő ágára lép a program, az fog lefutni. Tehát egy egyszerű kommentezéssel kiszedve a #define NORMAL_MODE-ot a teszteseteket futtathatjuk, egyébként a főprogramot.