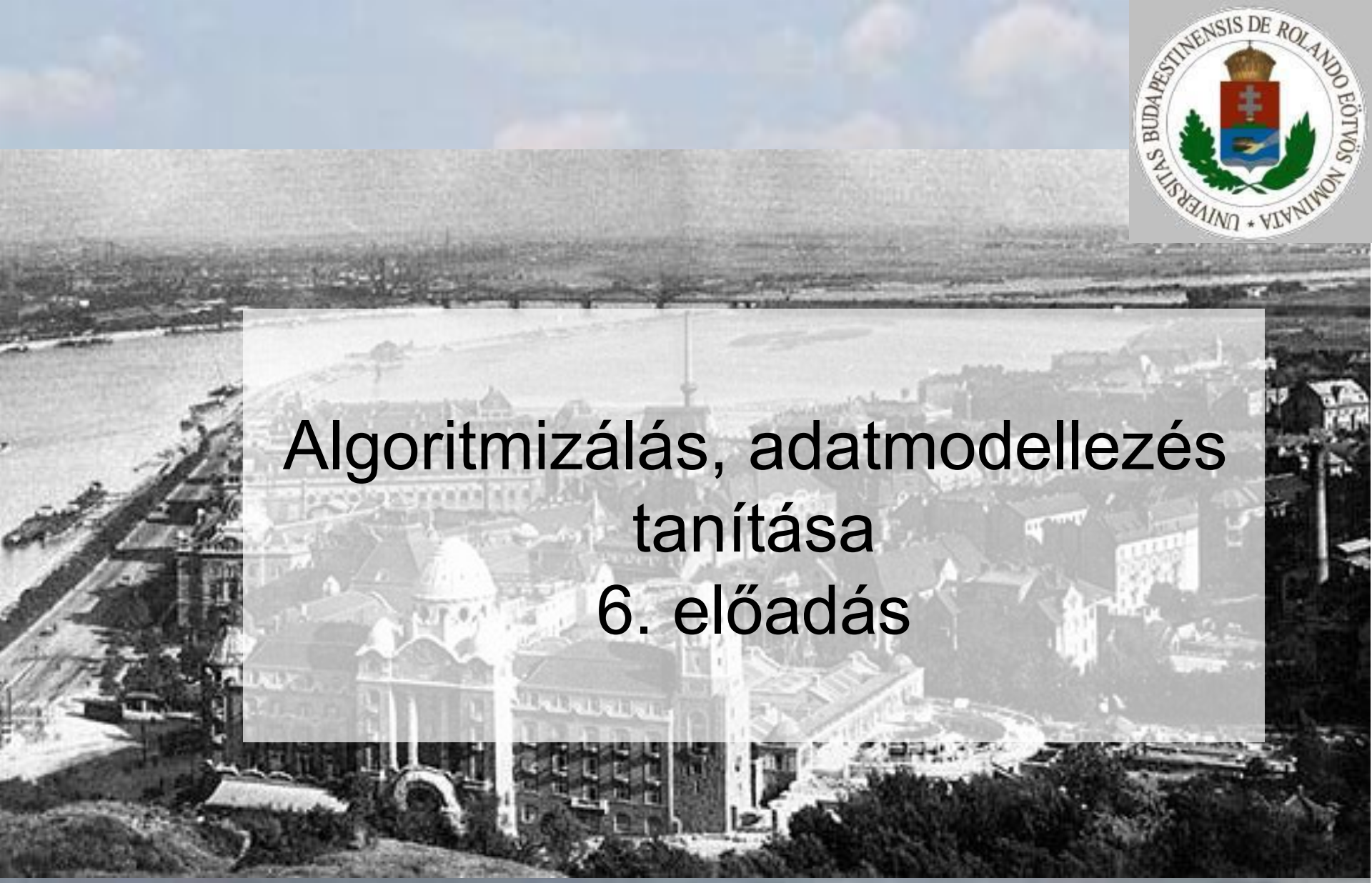
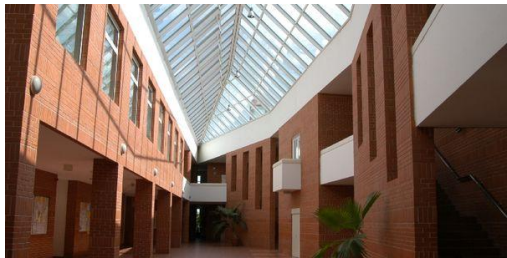


Algoritmizálás, adatmodellezés tanítása
6. előadás

Tesztelés, hibakeresés

Tesztelési módszerek

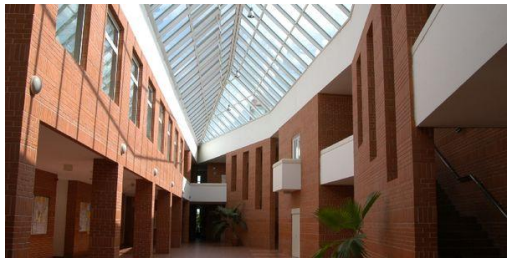
- *statikus tesztelés*
  - *kódellenőrzés*
  - *szintaktikus ellenőrzés*
  - *szemantikus ellenőrzés*
- dinamikus tesztelés
  - fekete doboz módszerek
  - fehér doboz módszerek



Tesztelés, hibakeresés

Dinamikus tesztelés

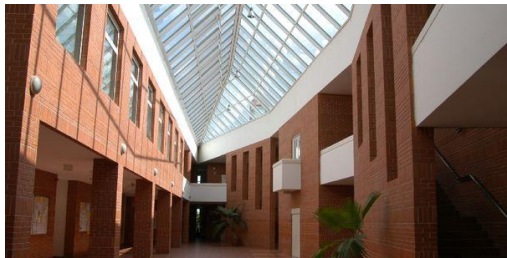
- *Fekete doboz* módszerek (nincs *kimerítő bemenet* – nem lehet minden lehetséges bemenetre kipróbálni): a teszteseteket a program specifikációja alapján választjuk.
- *Fehér doboz* módszerek (nincs *kimerítő út* – nem lehet minden végrehajtási sorrendre kipróbálni): a teszteseteket a program struktúrája alapján választjuk.



Tesztelés, hibakeresés

Fekete doboz módszerek

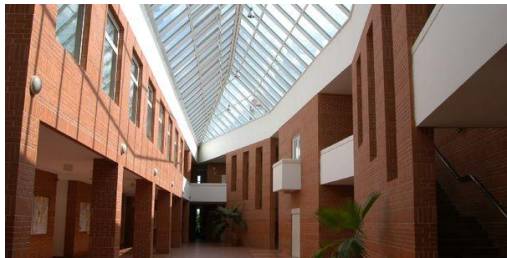
- **Ekvivalencia-osztályok módszere**: a bemeneteket (vagy a kimeneteket) soroljuk olyan osztályokba, amelyekre a program várhatóan egyformán működik; ezután osztályonként egy tesztesetet válasszunk!
- **Határeset elemzés módszere**: az ekvivalencia-osztályok határáról válasszunk tesztesetet!
- **Ok-hatás analízis**: bemeneti feltételek (okok) és kimeneti feltételek (hatások) kombinálása.
- **Véletlenszerű adatok generálása.**



Tesztelés, hibakeresés

Ekvivalencia osztályok módszere

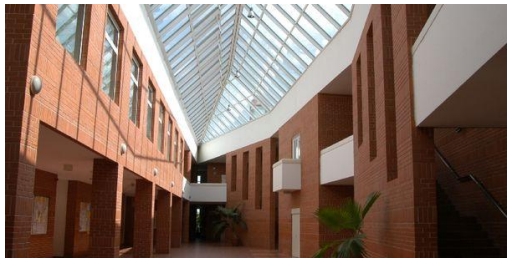
- Ha a bemeneti feltétel értéktartományt definiál, az érvényes ekvivalencia osztály legyen a megengedett bemenő értékek halmaza, az érvénytelen ekvivalencia osztályok pedig az alsó és a felső határoló tartomány. Pl. ha az adatok osztályzatok (értéküket 1 és 5 között van), akkor ezek az ekvivalencia osztályok rendre: $\{1 \leq i \leq 5\}$, $\{i<1\}$ és $\{i>5\}$.
- Ha a bemeneti feltétel értékek számát határozza meg, akkor az előzőhöz hasonlóan járjunk el. Pl. ha be kell olvassunk legfeljebb 6 karaktert, akkor az érvényes ekvivalencia osztály: 0-6 karakter beolvasása, az érvénytelen ekvivalencia osztály: 6-nál több karakter beolvasása. (0-nál kevesebb nem fordulhat elő.)



Tesztelés, hibakeresés

Ekvivalencia osztályok módszere

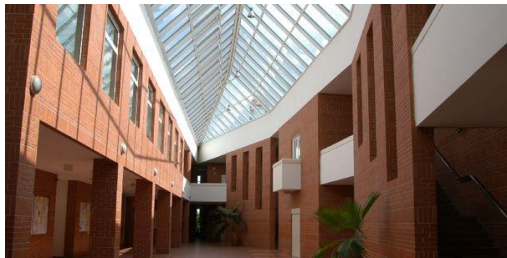
- Ha a bemenet feltétele azt mondja ki, hogy a bemenő adatnak valamilyen meghatározott jellemzővel kell rendelkezni, akkor két ekvivalencia osztályt kell felvenni: egy érvényeset és egy érvénytelent.
- Ha okunk van feltételezni, hogy a program valamelyik ekvivalencia osztályba eső elemeket különféleképpen kezeli, akkor a feltételezésnek megfelelően bontsuk az ekvivalencia osztályt további osztályokra.
- Alkalmazzuk ugyanezeket az elveket a kimeneti ekvivalencia osztályokra is!



Tesztelés, hibakeresés

A teszteseteket a következő két elv alapján határozhatjuk meg:

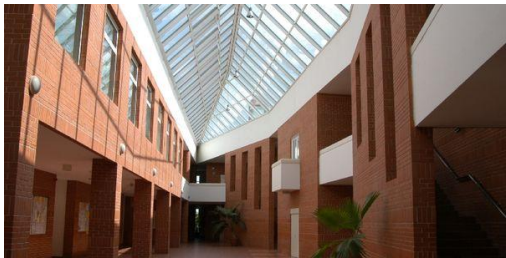
- Amíg az érvényes ekvivalencia osztályokat le nem fedtük, addig készítsünk olyan teszteseteket, amelyek minél több érvényes ekvivalencia osztályt lefednek!
- Minden érvénytelen ekvivalencia osztályra írjunk egy-egy, az osztályt lefedő tesztesetet. Több hiba esetén ugyanis előfordulhat, hogy a hibás adatok lefedik egymást, a második hiba kijelzésére az első hibajelzés miatt már nem kerül sor.



Tesztelés, hibakeresés

Határeset elemzés módszere

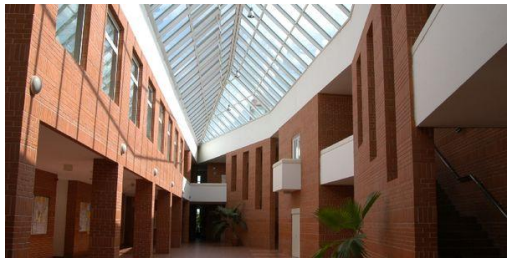
- Ha a bemeneti feltétel egy értéktartományt jelöl meg, írjunk teszteseteket az érvényes tartomány alsó és felső határára és az érvénytelen tartománynak a határ közelébe eső elemére! Pl.: ha a bemeneti tartomány a (0,1) nyílt intervallum, akkor a 0, 1, 0.01, 0.99 értékekre érdemes kipróbálni a programot.
- Ha egy bemeneti feltétel értékek számosságát adja meg, akkor hasonlóan járjunk el, mint az előző esetben. Pl.: ha rendeznünk kell 1-128 nevet, akkor célszerű a programot kipróbálni 0, 1, 128, 129 névvel.



Tesztelés, hibakeresés

Fehér doboz módszerek

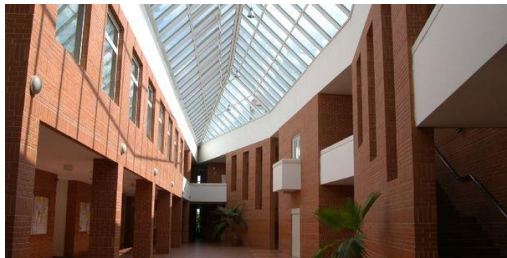
- egy kipróbálási stratégiát választunk a program szerkezete alapján,
- a stratégia alapján megadott teszt-utakhoz teszt-predikátumokat rendelünk,
- a tesztpredikátumok ekvivalencia osztályokat jelölnek ki, amelyekből egy-egy tesztesetet választunk.



Tesztelés, hibakeresés

Fehér doboz módszerek – kipróbálási stratégiák

- **utasítás lefedés**: minden utasítást legalább egyszer hajt-sunk végre!
- **feltétel lefedés**: minden feltétel legyen legalább egyszer igaz, illetve hamis!
- **részfeltétel lefedés**: minden részfeltétel legyen legalább egyszer igaz, illetve hamis!

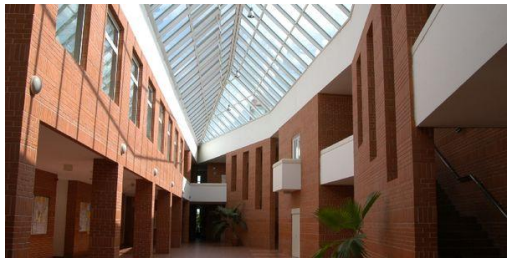


Tesztelés, hibakeresés

Fehér doboz módszerek – teszteset generálás

Bázisútnak nevezzük a programgráf olyan útját, amely

- a kezdőponttól a legelső elágazás- vagy ciklusfeltétel kiértékeléséig tart,
- elágazás- vagy ciklusfeltételtől a következő elágazás- vagy ciklusfeltétel helyéig vezet,
- elágazás- vagy ciklusfeltételtől a program végéig tart, s közben más feltétel kiértékelés nincs.



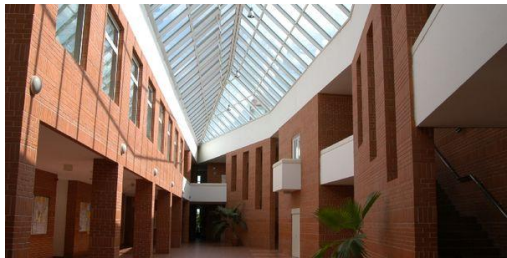
Tesztelés, hibakeresés

Fehér doboz módszerek – teszteset generálás

Tesztutaknak nevezzük a programgráfon átvezető, a kezdőponttól a végpontig haladó olyan utakat, amelyek minden bennük szereplő élt pontosan egyszer tartalmaznak.

Tesztpredikátumnak nevezzük azokat a bemenő adatokra vonatkozó feltételeket, amelyek teljesülése esetén pontosan egy tesztúton kell végighaladni.

A teszteset generálás első lépése tehát a minimális számú olyan tesztút meghatározása, amelyek lefedik a kipróbálási stratégiának megfelelően a programgráfot.

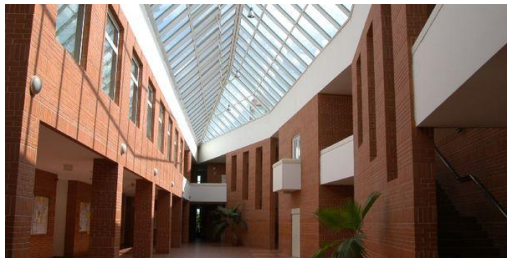


Tesztelés, hibakeresés

Fehér doboz módszerek – teszteset generálás

A tesztpredikátum előállítása:

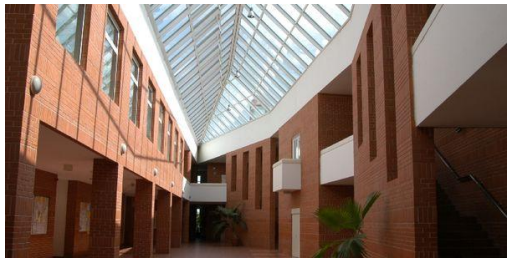
Ehhez a program **szimbolikus végrehajtás**ára van szükség. Induljunk ki az előfeltételből! Haladjunk a programban az első elágazás- vagy ciklusfeltételig, s a formulát a közbülső műveleteknek megfelelően transzformáljuk! A tesztútnak megfelelő ág feltételét **és** kapcsolattal kapcsoljuk hozzá a tesztpredikátumhoz, majd folytassuk a szimbolikus végrehajtást egészen a program végpontjáig!



Tesztelés, hibakeresés

Hibakeresési eszközök

- Memória-, változó-kiírás
- Nyomkövetés (hibától visszafelé is)
- Adat-nyomkövetés
- Állapot-nyomkövetés (pl. paraméterek)
- Töréspont elhelyezése
- Lépésenkénti végrehajtás
- A hiba helyének és okának kijelzése
- Speciális ellenőrzések (pl. indexhatár)



Tesztelés, hibakeresés

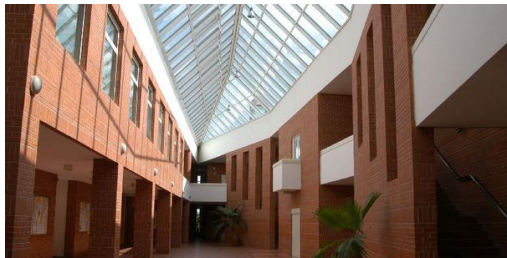
Hibakeresési módszerek

Célja:

- *A bemenetnek mi az a része, amire hibásan működik a program?*
- *Hol található a programban a hibát okozó utasítás?*

Módszerfajták:

1. Indukciós módszer (hibásak körének bővítése)
2. Dedukciós módszer (hibásak körének szűkítése)
3. Hibakeresés hibától visszafelé

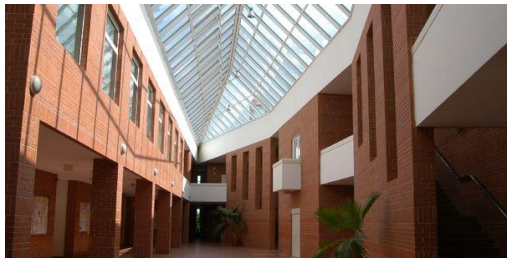


Helyességbizonyítás

Helyességbizonyítás

Egy U program az x bemenő adatokra felírt ef(x) előfeltétel, valamint az uf(x,y) utófeltétel mellett **parciálisan** (részlegesen) **helyes**, ha ∀x-re, amelyre a program futása befejeződik: ef(x) → uf(x,y).

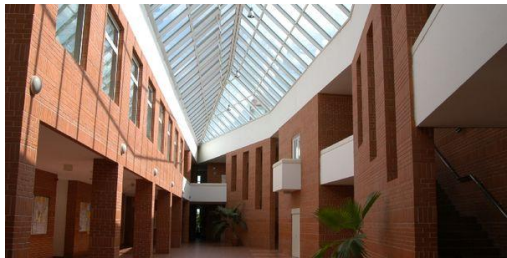
Jelölése: {ef(x)} U(x,y) {uf(x,y)}.



Helyességbizonyítás

Egy U program az x bemenő adatokra felírt ef(x) előfeltétel, valamint az uf(x,y) utófeltétel mellett **befejeződik**, ha ∀x-re: ef(x) → a program eljut a végpontjához.

Egy U program az x bemenő adatokra felírt ef(x) előfeltétel, valamint az uf(x,y) utófeltétel mellett **teljesen helyes**, ha parciálisan helyes és befejeződik.



Helyességbizonyítás

Axiómák

1. Az értékadás axiómája:

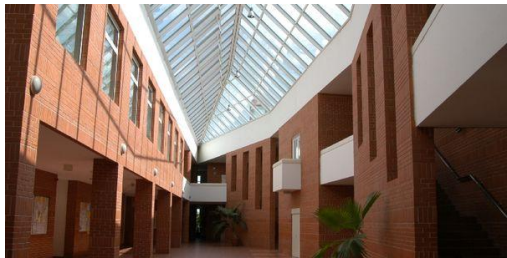
$$\{ef(x)\}\ y:=f(x)\ \{ef(x) \text{ és } y=f(x)\}$$

$$\{ef(x,f(x))\}\ y:=f(x)\ \{uf(x,y)\}$$

2. Az üres utasítás axiómája:

$$\{ef(x)\}\ \text{üres}\ \{ef(x)\}$$

.



Helyességbizonyítás

Következtetési szabályok

1. Szekvencia bizonyítása: $U=U_1; U_2$

$\{ef(x)\}\ U_1\ \{P(x)\}$ és $\{P(x)\}\ U_2\ \{uf(x)\}$

$\rightarrow \{ef(x)\}\ U_1; U_2\ \{uf(x)\}$

2. Elágazás bizonyítása:

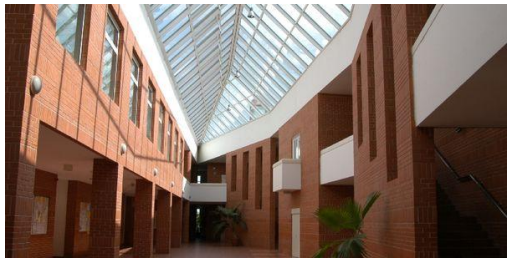
U= Ha P(x) akkor U_1 különben U_2

$\{ef(x)$ és $P(x)\}\ U_1\ \{uf(x)\}$ és

$\{ef(x)$ és nem $P(x)\}$ U2 $\{uf(x)\} \rightarrow$

$\{ef(x)\}$ Ha P(x) akkor U_1

különben $U_2\ \{uf(x)\}$



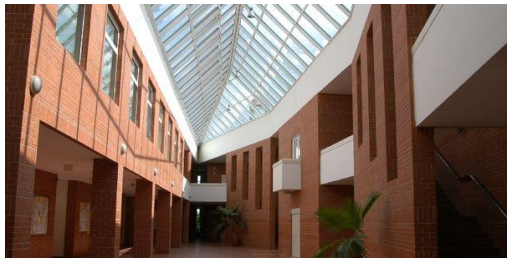
Helyességbizonyítás

Következtetési szabályok

3. Ciklus bizonyítása: U= Ciklus amíg cf(x)
Cm(x)
Ciklus vége

Definiáljunk a ciklushoz egy I(x) invariáns állítást!

ef(x) → I(x) és {I(x) és cf(x)} Cm(x) {I(x)} és
(I(x) és nem cf(x)) → uf(x)
→ {ef(x)} Ciklus amíg cf(x)
Cm(x)
Ciklus vége {uf(x)}



Helyességbizonyítás

Következtetési szabályok

4. Ciklus befejeződése: U= Ciklus amíg cf(x)

Cm(x)

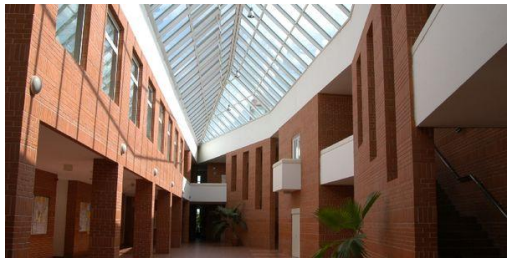
Ciklus vége

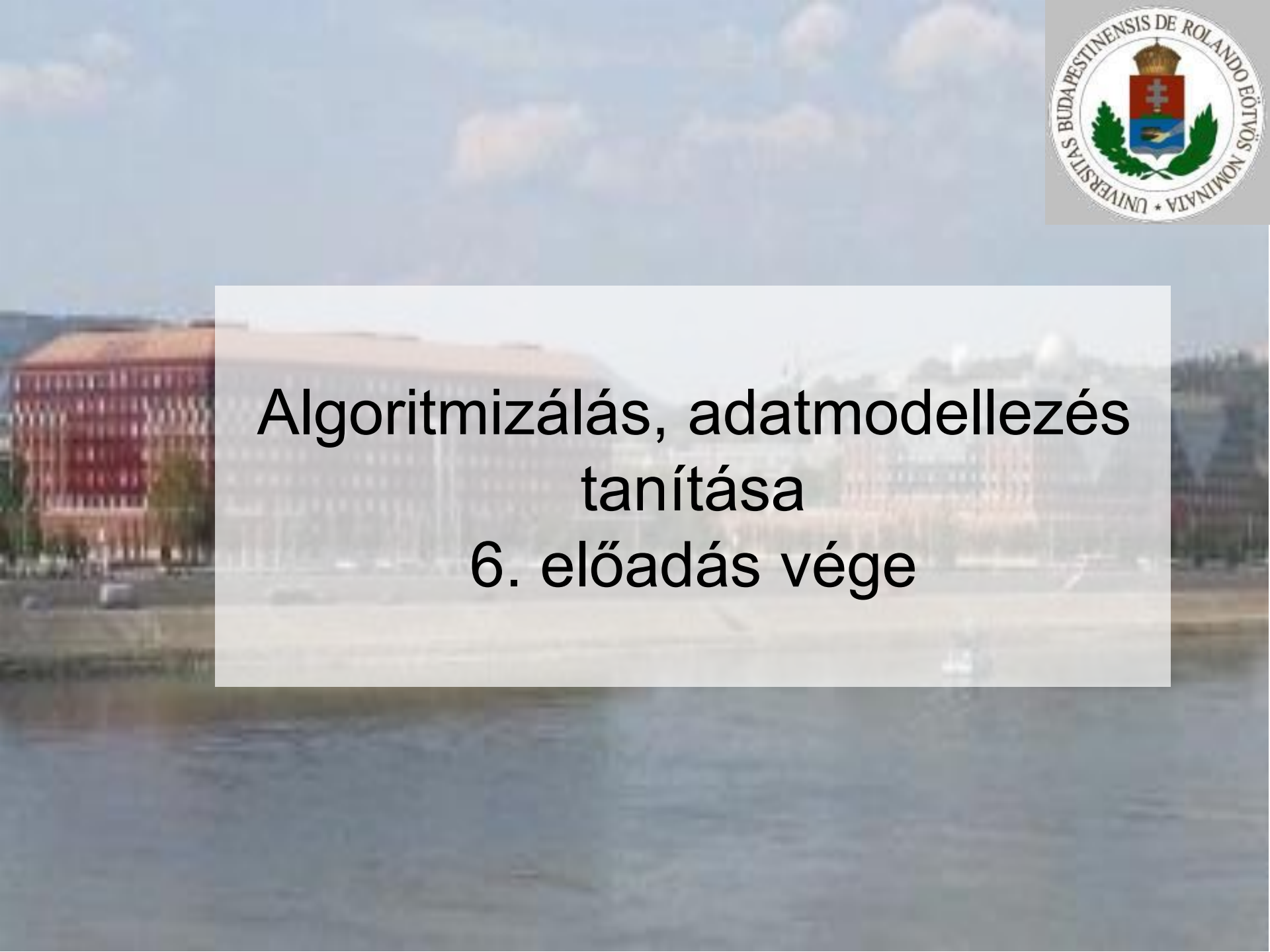
Definiáljunk a ciklushoz egy t(x) variáns függvényt!

$$I(x) \rightarrow t(x) \in \mathcal{N} \text{ és}$$

$$\{I(x) \text{ és } cf(x) \text{ és } t(x)=t_0\}\ Cm(x)\ \{t(x)<t_0\}$$

$$\rightarrow \text{a ciklus befejeződik}$$





Algoritmizálás, adatmodellezés tanítása
6. előadás vége