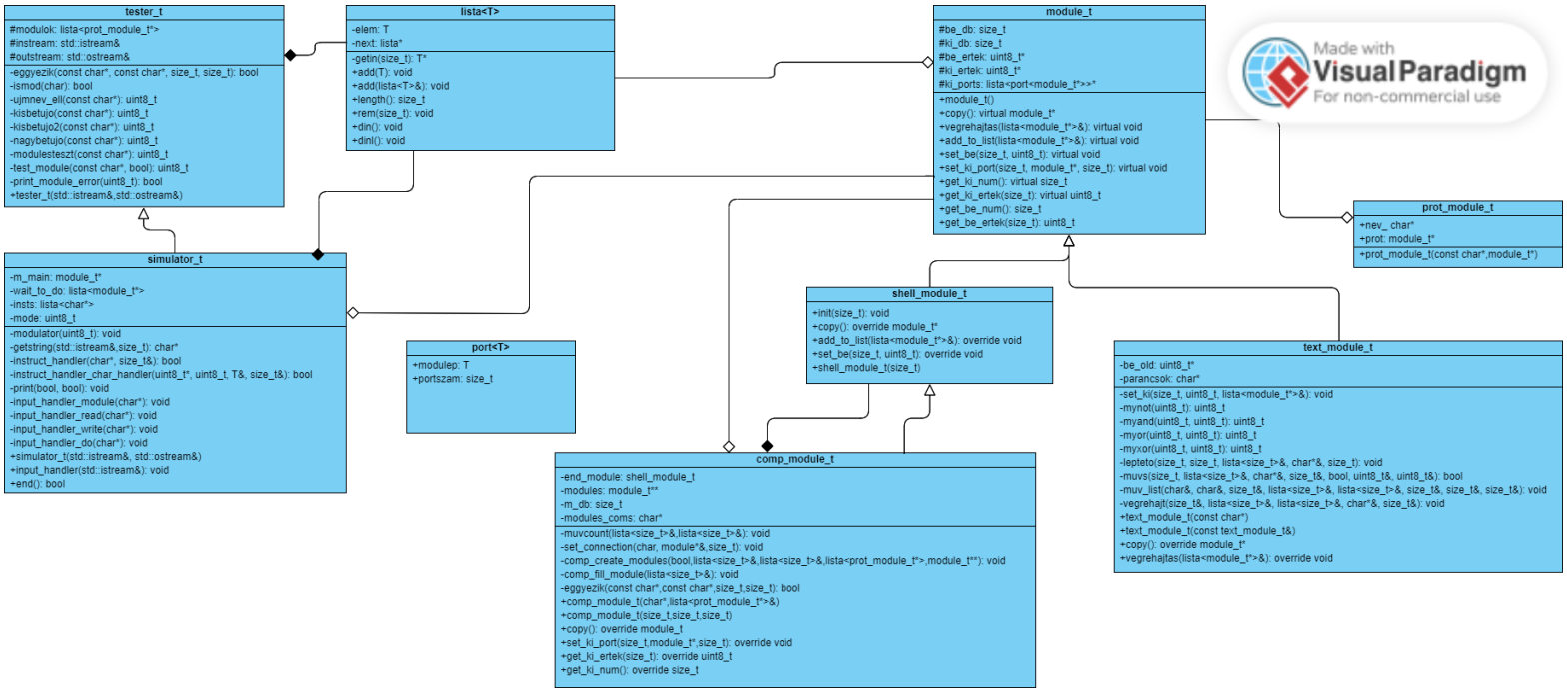
Logikai modulok szimulálása konzolon

Az UML Diagram a VisualParadigm online felületén készült.



Classok és fontosabb(nem maguktól érthetődő) függvényeik:

1. lista:

láncolt lista, aminek lezáró elemének next-je nullptr, ebben már nem lesz használt adat.

1.1. din: a listában tárolt elemek törlése

1.2. dinl: a listában tárolt tömbök törlése

1.3. rem: remove i-edik elem

2. module_t:

tárolja az input és output értékeket, valamint, hogy milyen modulok vannak rákötve melyik portjukkal

2.1. copy: visszatér egy a modulról készült másolat pointerével

2.2. vegrehajtas: semmit nem csinál

2.3. add_to_list: hozzáadja magát a paraméterként kapott listához.

2.4. set_be: a megadott indexű inputra beírja a megadott értéket.

2.5. set_ki_port: a megadott indexű outputra rákötí a kapott modul kapott indexű bemenetét

2.6. get_ki_num: visszatér, hogy hány outputja van a modulnak

2.7. get_ki_ertek: visszatér az indexedik output értékével

2.8. get_be_num: visszatér, hogy hány inputja van a modulnak

2.9. get_be_ertek: visszatér az indexedik input értékével

3. text_module_t:

ez az a modul, ahol számolások is folynak

3.1. végrehajtás: lefutatja az outputokhoz tartozó kiértékeléseket és ha változott az output értéke, akkor a hozzá kapcsolt modulokat hozzáadja a várakozók listájához:

3.1.1. Egy outputhoz tartozó szöveg kimásolása, inputok betűi helyére értékek behelyettesítése.(vegrehajt)

3.1.2. Műveleti lista létrehozása, műveleti sorrenddel.(muv_list)
3.1.3. Majd ebben a sorrendben, műveletek elvégzése, szintén segédfüggvényekkel(muvs), a kimásolt szöveg folyamatosan változik, minden végrehajtott művelet helyére az eredmény kerül (lepteto), és így a végén kijön egy 1 karakteres végeredmény.
3.1.4. Ezek után, ha változott az output akkor a hozzá kapcsolt moduloknak megváltoztatja a bemeneti adatait és a modulokat hozzáadja a várakozók listájához(set_ki)

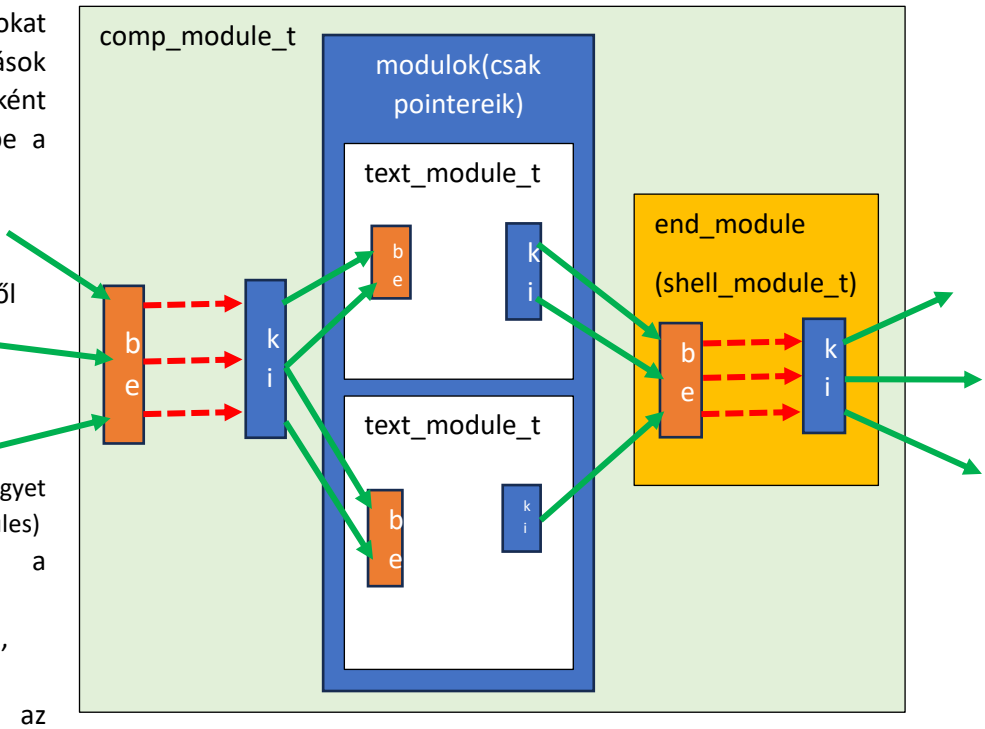
4. shell_module_t:

ez a modul vázat biztosít további moduloknak

4.1. init: portok számának megadása, létrehozza a tömböket
4.2. add_to_list: átírva, a kapott indexű outputjára kötött modulokat adja hozzá a várakozók listájához.
4.3. set_be: a saját be és kimenetét beállítja az adott indexen az adott értékre, valamint a hozzá kapcsolt moduloknak megváltoztatja a bemeneti adatait(a cél modul set_be hívásával)

5. comp_module_t:

ez a modul, ami modulokat tartalmaz, és a kapcsolások leírását, valamint lezárásként egy shell_module_t-t, amibe a belső modulok futhatnak



5.1. comp_module_t létrehozásakor: Feltérképezi mettől meddig tartanak a modulok(muvcount)
5.1.1. Létrehoz minden használt modulból egyet (comp_create_modules)
5.1.2. Majd elvégzi a bekötéseket (comp_fill_module, set_connection)
5.2. set_ki_port: az end_moduljának hívja meg a set_ki_port függvényét
5.3. get_ki_ertek: az end_moduljának hívja meg a get_ki_ertek függvényét
5.4. get_ki_num: az end_moduljának hívja meg a get_ki_num függvényét

6. prot_module_t:

6.1. modulok és neveik tárolására való struct, konstruktorral és destruktorral a char* kezelése miatt

7. port:

7.1. generikus struct modulhivatkozás és portjának tárolására

8. tester_t:

8.1. tesztelésekre használható, tárolja a ki és bemeneti streamet, valamint a már jó modulokat
8.2. eggyezik: stringek összehasonlítására használt függvény

8.3. ismod: megnézi, hogy a karakter egy a végrehajtásnál használható karakter-e
8.4. ujmnev_ell: megnézi, hogy megfelelő-e az új modul név: nem volt még használva, nem tartalmaz tiltott karaktereket, '_'-vel kezdődik
8.5. kisbetujo: text module esetén megnézi, hogy jó-e a szintaxis, nem tartalmaz tiltott karaktereket, jó-e a bekötés
8.6. kisbetujo2: comp module esetén megnézi, hogy jó-e a szintaxisa az io-nak, jó-e a bekötésük
8.7. nagybetujo: comp module esetén megnézi, hogy jó-e a szintaxisa a belső vezetékezésnek, jó-e a bekötésük
8.8. modulesteszt: megnézi, hogy a használni kívánt modul létezik e már és megfelelő számú input és output van e megadva
8.9. test_module: a fentebbi függvényekkel teszteli, hogy létrehozható-e a kívánt új modul, és ha igen hozzáadja a modulok listájához
8.10. print_module_error: kiírja az outstreamre a hibakódnak megfelelő hibát

9. simulator_t:

9.1. modulator: megváltoztatja a modulátorok beállítását
9.2. print: kiírja a m_main module be és ki értékeit
9.3. `input`_handler_do: végrehajtás utasítást hajt végre, először értelmezi (instruct_handler_do, instruct_handler_char_handler) segítségével
9.3.1. ha a lefutások száma > 0 akkor végigmegy a várakozó modulok listáján és meghívja azok végrehajtás függvényüket, az ezek miatt triggerelt modulok egy új várakozó modulok listába mentődnek. amit a régi listáról minden végrehajtódik, az új lista veszi át a régi helyét
9.3.2. ez a folyamat a lefutások számaszor hajtódik végre.
9.4. input_handler_module: létrehozza az új modult, ha lehet(test_module), és ha az új modul neve '_main' akkor példányosodik
9.5. input_handler_read: fájlból hajtja végig az utasításokat
9.6. input_handler_write: végigmegy az insts listán(kiadott utasítások) és megkérdezi, hogy melyeket akarjuk menteni a fájlba, és közben menti azokat.
9.7. `input_handler`: az előző négy függvényt kezelő függvény,
9.7.1. ha az utasítás '_'-vel kezdődik: input_handler_module,
9.7.2. '<'-vel kezdődik: input_handler_read,
9.7.3. '>'-vel kezdődik: input_handler_write,
9.7.4. különben input_handler_do
9.8. end: ha end modulátor volt, akkor true-val tér vissza különben false-al