OEP 3. beadandó doksi – BMA0B9 Molnár Dániel

Feladat leírás:

A hobbi állatoknak az életkedvük megőrzéséhez a táplálékon túl egyéb dolgokra is szükségük van: a halaknak oxigén dús, megfelelő hőmérsékletű vízre; a madaraknak tágas, tiszta kalitkára; a kutyáknak rendszeres foglalkoztatásra. Pisti számos hobbi állatot tart: halakat, madarakat és kutyákat. Állatainak van neve és ismerhető az életkedvüket mutató 0 és 100 között szám (0 esetén az állat elpusztul). Pistinek vannak jobb és rosszabb napjai. Mikor nagyon jó kedvű, egyik állatáról sem feledkezik meg: ilyenkor a halak életkedve 1-gyel, a madaraké 2-vel, a kutyáké 3-mal nő. Átlagos napokon csak a kutyáival foglalkozik, a többi állat életkedve ilyenkor csökken: a halaké 3-mal, a madaraké 1-gyel. Amikor rosszkedvű, csak a legszükségesebb teendőket látja el és ezért minden állat egy kicsit szomorúbb lesz: a halak 5 egységgel, a madarak 3-mal, a kutyák 10-zel. Pisti kedve egy adott napon egy kategóriával jobb lesz attól, ha minden élő kedvencének az életkedve legalább 5. Nevezze meg a legszomorúbb (legkisebb az életkedvű) állatot, amelyik még nem pusztult el a vizsgált napok után! Ha több ilyen életkedvű állat is létezik, akkor írja ki az összesnek a nevét! Naponként mutassa meg az állatok összes tulajdonságát! Az állatok adatait egy szöveges állományban találjuk. Az első sor tartalmazza az állatok számát, amelyet külön-külön sorban az állatok adatai követnek. Ebben egy karakter azonosítja az állat fajtáját (H – hal, M – madár, K – kutya), amit szóköz után az állat neve követ, majd újabb szóköz után a kezdeti életkedve. Az állományban az állatok felsorolását követő utolsó sorban egy betű sorozat (sztring) írja le Pisti kedvének az egymás utáni napokon való alakulása: j – jó kedvű, a – átlagos, r – rosszkedvű. Feltehetjük, hogy a fájl formátuma helyes.

Program indítása:

Cmake-build-debug mappában az beadando3.exe-t kell futtatni, filet nem lehet megadni, defaultból a test.txt-t dolgozza fel, igy ha más fileal szeretné kipróbálni, kérem írja felül.

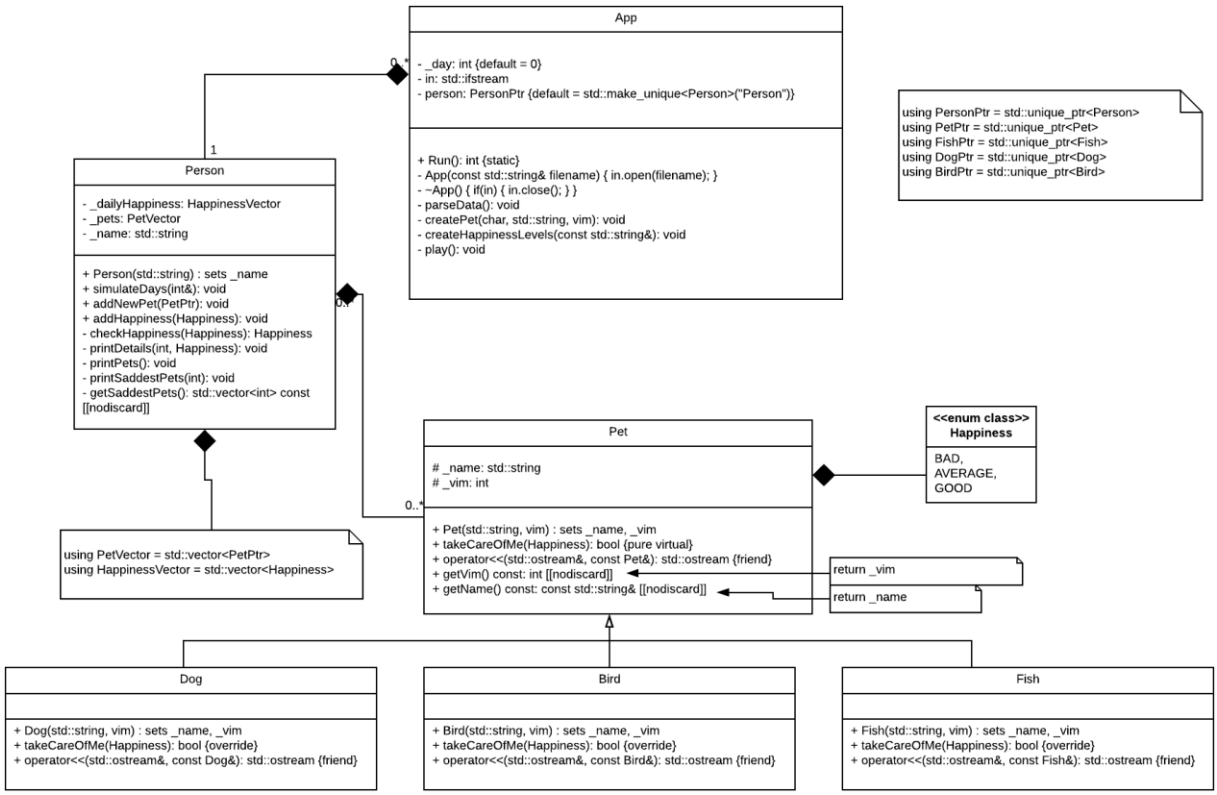
Arra az esetre, ha nincs meg a binary kérem cmakel fordítsa le.

<path_to_cmake.exe> --build <path>\beadando3\cmake-build-debug --target beadando3 -- -j 6

Terv:

Kelleni fog az állatoknak egy Base class, illetve egy Person class, aki birtokolja az állatokat (akármennyit). Illetve a szépség kedvéért egy App class amiben parseoljuk az adott filet és megoldjuk a feladatokat.

UML diagram:



Érdekesebb algoritmusok:

```
Happiness Person::checkHappiness(Happiness dailyHappiness)
FUNCTION
    IF (dailyHappiness != Happiness::GOOD) THEN
        increaseHappiness = true;
        FOR (pet in pets)
            IF (pets vim < 5) THEN
                increaseHappiness = false;
                break;
            FI
        END
        IF increaseHappiness THEN
            Update dailyHappiness value to the next level
        FI
    FI
    RETURN dailyHappiness;
END
---------------------------------
void Person::simulateDays(int& day)

FUNCTION
    FOR (currentHappiness in _dailyHappiness)
        happiness = checkHappiness(currentHappiness);
        FOR (iterate through pets)
            IF (take care of pet is unsuccesful) THEN
                Erase pet
            ELSE
                increment iterator
            FI
        END
```

```
        Print details of person on the given day, then increment day
    END
    Print saddest pets here
END
--------------------------------
std::vector<int> Person::getSaddestPets() const
FUNCTION
    Declare vector that will store pet’s index
    min = 100;
    FOR (iterate throught pets w index)
        IF (pets[i]’s vim <= min) THEN
            IF (pets[i]’s vim != min) THEN
                min = pets[i]’s vim
                IF (vector is not empty) THEN
                    Clear vector
                FI
            FI
            Add pets index to vector;
        FI
    END
    RETURN vector;
END
--------------------------------
int App::Run()

FUNCTION

    filename = "../test.txt";

    App app(filename);

    app.parseData();

    app.play();

    RETURN 0;

END
--------------------------------
void App::parseData()
FUNCTION
    std::string line;
    int counter = 0;
    std::string happiness;
    std::getline(in, line);
    std::istringstream iss(line);
    iss >> counter;
    //parse content line by line
    WHILE (std::getline(in, line)) {
        iss = std::istringstream(line);
        IF(counter-- > 0) THEN
            char species;
            std::string name;
            int vim = 0;
            iss >> species >> name >> vim;
            createPet(species, name, vim);
        ELSE
            iss >> happiness;
            createHappinessLevels(happiness);
        FI
    END
END
--------------------------------
```

```
void App::createPet(char species, std::string name, int vim)
FUNCTION
    IF (species == 'H') THEN
        auto newPet = std::make_unique<Fish>(std::move(name), vim);
        person->addNewPet(std::move(newPet));
    ELSE IF (species == 'M') THEN
        auto newPet = std::make_unique<Bird>(std::move(name), vim);
        person->addNewPet(std::move(newPet));
    ELSE
        auto newPet = std::make_unique<Dog>(std::move(name), vim);
        person->addNewPet(std::move(newPet));
    FI
END
--------------------------------
void App::createHappinessLevels(const std::string &happiness)
FUNCTION
    FOR (const auto &daily : happiness)
        SWITCH (daily) {
            case 'j':
                person->addHappiness(Happiness::GOOD);
                break;
            case 'a':
                person->addHappiness(Happiness::AVERAGE);
                break;
            case 'r':
                person->addHappiness(Happiness::BAD);
                break;
            default:
                std::cout << "undefined happiness level: " << daily << std::endl;
        FI
    END
END
--------------------------------
```

Teszteset:

Több teszt egy tesztben:

Test.txt fájlban többféle állat is meg van adva, illetve Pisti kedve egy hosszabb stringként.

Helyesen változik mindene állatfajtának a kedve Pisti kedvétől függően.