Stooge Sort (Придурковатая сортировка) + компоненты калькулятора

1. Общая постановка задачи

Необходимо в ранее созданный компонент используя приемы программирования включение/агрегирование добавить компоненты калькулятора, выполняющие операции сложения, вычитания, деления и умножения.

2. Реализация

Приемы программирования через включение и агрегирование представляют собой два подхода к созданию составных объектов:

- Включение - когда объект является частью другого объекта, и их жизненные циклы связаны (при уничтожении контейнера уничтожается и включенный объект)
- Агрегирование - когда объекты связаны слабее, их жизненные циклы независимы

```
static int16_t ECOCALLMETHOD CEcoLab1_QueryInterface(/* in */ IEcoLab1Ptr_t me, /* in */ const UGUID* riid, /* out */ void** ppv) {
    CEcoLab1* pCMe = (CEcoLab1*)me;

    /* Проверка указателей */
    if (me == 0 || ppv == 0) {
        return ERR_ECO_POINTER;
    }

    /* Проверка и получение запрошенного интерфейса */
    if ( IsEqualUGUID(riid, &IID_IEcoLab1) ) {
        *ppv = &pCMe->m_pVTblIEcoLab1;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*)pCMe);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoUnknown)) {
        *ppv = &pCMe->m_pVTblIEcoLab1;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX) && pCMe->m_pInnerUnknown != 0) {
        return pCMe->m_pInnerUnknown->pVTbl->QueryInterface(pCMe->m_pInnerUnknown, riid, ppv);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX)) {
        *ppv = &pCMe->m_pVTblIEcoCalculatorX;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoCalculatorY)) {
        *ppv = &pCMe->m_pVTblIEcoCalculatorY;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    }
    else {
        *ppv = 0;
        return ERR_ECO_NOINTERFACE;
    }
    return ERR_ECO_SUCCESES;
}
```

```
static int16_t ECOCALLMETHOD CEcoLab1_IEcoCalculatorX_QueryInterface(/* in */ struct IEcoCalculatorX* me, /* in */ const UGUID* riid, /* out */ void** ppv) {
    CEcoLab1* pCMe = (CEcoLab1*) ((uint64_t) me - sizeof(struct IEcoLab1*));

    /* Проверка указателей */
    if (me == 0 || ppv == 0) {
        return -1;
    }

    /* Проверка и получение запрошенного интерфейса */
    if (IsEqualUGUID(riid, &IID_IEcoLab1)) {
        *ppv = &pCMe->m_pVTblIEcoLab1;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoUnknown)) {
        *ppv = &pCMe->m_pVTblIEcoLab1;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX) && pCMe->m_pInnerUnknown != 0) {
        return pCMe->m_pInnerUnknown->pVTbl->QueryInterface(pCMe->m_pInnerUnknown, riid, ppv);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX)) {
        *ppv = &pCMe->m_pVTblIEcoCalculatorX;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    }
    else if (IsEqualUGUID(riid, &IID_IEcoCalculatorY)) {
        *ppv = &pCMe->m_pVTblIEcoCalculatorY;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    } else {
        *ppv = 0;
        return -1;
    }
    return 0;
}

static int16_t CEcoLab1_IEcoCalculatorY_QueryInterface(/* in */ struct IEcoCalculatorY* me, /* in */ const UGUID* riid, /* out */ void** ppv) {
    CEcoLab1* pCMe = (CEcoLab1*) ((uint64_t) me - sizeof(struct IEcoLab1*) * 2);

    /* Проверка указателей */
    if (me == 0 || ppv == 0) {
        return -1;
    }

    /* Проверка и получение запрошенного интерфейса */
    if (IsEqualUGUID(riid, &IID_IEcoLab1)) {
        *ppv = &pCMe->m_pVTblIEcoLab1;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    } else if (IsEqualUGUID(riid, &IID_IEcoUnknown)) {
        *ppv = &pCMe->m_pVTblIEcoLab1;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    } else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX) && pCMe->m_pInnerUnknown != 0) {
        return pCMe->m_pInnerUnknown->pVTbl->QueryInterface(pCMe->m_pInnerUnknown, riid, ppv);
    } else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX)) {
        *ppv = &pCMe->m_pVTblIEcoCalculatorX;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    } else if (IsEqualUGUID(riid, &IID_IEcoCalculatorY)) {
        *ppv = &pCMe->m_pVTblIEcoCalculatorY;
        pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*) pCMe);
    } else {
        *ppv = 0;
        return -1;
    }

    return 0;
}
```

```
/* Сохранение указателя на системный интерфейс */
pCMe->m_pISys = (IEcoSystem1*) pIUnkSystem;

/* Получение интерфейса для работы с интерфейсной шиной */
result = pCMe->m_pISys->pVTbl->QueryInterface(pCMe->m_pISys, &IID_IEcoInterfaceBus1, (void**) &pIBus);

/* Проверка указателей */
if (me == 0) {
    return result;
}

/* Сохранение указателя на системный интерфейс */
pCMe->m_pISys = (IEcoSystem1*) pIUnkSystem;

result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorE, 0, &IID_IEcoCalculatorY, (void**) &pCMe->m_pIEcoCalculatorY);
if (result != 0 || pCMe->m_pIEcoCalculatorY == 0) {
    result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorD, 0, &IID_IEcoCalculatorY, (void**) &pCMe->m_pIEcoCalculatorY);
}
if (result != 0) {
    goto release;
}

result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorB, pOuterUnknown, &IID_IEcoUnknown, (void**) &pCMe->m_pInnerUnknown);
if (result != 0 || pCMe->m_pInnerUnknown == 0) {
    result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorA, 0, &IID_IEcoCalculatorX, (void**) &pCMe->m_pIEcoCalculatorX);
}
```

3. Пример работы

Была продемонстрирована работа всех операций сложения, вычитания, деления и умножения. Затем проверяется доступность в получении каждого интерфейса из каждого интерфейса. В конце пользователь имеет возможность запустить тесты для сортировок.

```
C:\Windows\system32\cmd.exe
A/B components tests:
36 + 2 = 38
36 - 2 = 34

D/E components tests:
36 * 2 = 72
36 / 2 = 18

IEcoLab1 -> pIEcoCalculatorX [OK]
IEcoLab1 -> pIEcoCalculatorY [OK]
IEcoLab1 -> IEcoLab1 [OK]
pIEcoCalculatorX -> pIEcoCalculatorY [OK]
pIEcoCalculatorX -> IEcoLab1 [OK]
pIEcoCalculatorX -> pIEcoCalculatorX [OK]
pIEcoCalculatorY -> pIEcoCalculatorX [OK]
pIEcoCalculatorY -> pIEcoCalculatorY [OK]
pIEcoCalculatorY -> IEcoLab1 [OK]


Next are sort tests. Wanna see? (y/n)
y

/// int32_t test ///
Test #1: 693 410 -372 -557 959 -936 -654 129 -265 -567 -484 -334 -644 -234 -987 188 768 -452 294 700 ...
Test took 4.000000 ms
Result #1: -988 -987 -977 -973 -971 -969 -968 -936 -860 -814 -765 -749 -722 -716 -691 -688 -672 -654 -644 -612 ...

Test #2: 238 -93 985 348 -967 324 -242 145 -770 691 -310 -940 -833 -179 954 -652 -569 -672 -521 -691 ...
Test took 346.000000 ms
Result #2: -998 -997 -988 -980 -967 -962 -961 -961 -960 -958 -958 -953 -944 -940 -938 -934 -930 -920 -920 -914 ...

Test #3: -826 -288 -635 404 311 -424 10 -658 364 356 -504 741 360 -240 711 -989 217 -549 544 -18 ...
Test took 1048.000000 ms
Result #3: -998 -997 -997 -995 -994 -991 -989 -985 -983 -982 -981 -979 -979 -977 -976 -975 -973 -972 -970 -969 ...

/// long test ///
Test #1: -338 908 -85 278 -447 191 357 -415 -52 -621 414 541 -790 232 -340 766 -962 310 789 -599 ...
Test took 5.000000 ms
```

4. Выводы

Реализованный компонент калькулятора демонстрирует правильное использование приемов программирования включения и агрегирования. Благодаря использованию QueryInterface, любой компонент может получить доступ к любому другому интерфейсу, что обеспечивает гибкость и расширяемость системы.

Реализация соответствует требованиям задания:

- Добавлены компоненты для всех арифметических операций
- Использованы приемы включения/агрегирования
- Продемонстрирована работа QueryInterface
- Показана возможность получения любого интерфейса из любого другого

Такой подход позволяет легко расширять функциональность системы, добавляя новые операции без изменения существующего кода, что соответствует принципу открытости/закрытости объектно-ориентированного программирования.