

Тема 6: Контейнеры

Цель

Получить навыки создания собственных контейнеров на базе существующих классов

Общее задание

- Определить класс коллекции для сохранения объектов в соответствии со своей прикладной областью
- В коллекции должен быть перегружен оператор []
- Класс должен выполнять сохранения данных в файл и дальнейшую загрузку из файла
- Загрузку и сохранение в файл осуществлять с помощью библиотеки FileStoreLibrary

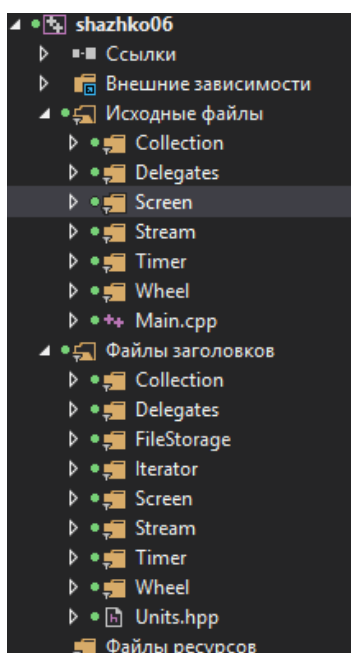
Прикладная область

- Колеса

Индивидуальное задание

- Коллекцию реализовать на основе списков

Структура проекта



Описание разработанных типов данных

Collection	
abstract	Абстрактный класс который задает интерфейс контейнеров
SimpleList	Класс коллекции, который реализует интерфейс ICollection<item>
Node	Структура, благодаря которой связываются элементы между собой и хранят пользовательские данные
SimpleListIterator	Класс итератор, благодаря которому можно делать обход коллекции
Delegate	
DelegateVoid	Класс для удобной работы с делегатами
IDelegateVoid	Задаёт интерфейс
MethodDelegateVoid	Класс реализующий интерфейс IDelegateVoid, для работы с функциями в классах
StaticDelegateVoid	Класс реализующий интерфейс IDelegateVoid, для работы со статическими функциями
Iterator	
abstract	Абстрактный класс который задает интерфейс итераторов
Screen	
ScreenCreator	
abstract	Абстрактный класс, который задает интерфейс фабричного метода для создания объектов наследующий класс ScreenCreator
CarWheelScreenCreator	Класс реализующий фабричный метод для создания объектов CarWheelScreen
DefaultScreenCreator	Класс реализующий фабричный метод для создания объектов DefaultScreen
GraphScreenCreator	Класс реализующий фабричный метод для создания объектов GraphScreen
abstract	Класс описывающий базовое отображение объектов
CarWheelScreen	Класс описывающий отображение объектов класса CarWheelScreen
DefaultScreen	Класс описывающий отображение объектов класса DefaultScreen
GraphScreen	Класс описывающий расширенное отображение объектов класса GraphScreen
Stream	
StreamHelper	
abstract	Абстрактного класса, который задает интерфейс ввода
SimpleInputStreamHelper	Вспомогательный класс для чтения параметров из потока
SimpleOutputStreamHelper	Вспомогательный класс для записи параметров в поток
SimpleStreamHelperFactory	Класс реализующий интерфейс абстрактной фабрики, для создания вспомогательных объектов ввода-вывода параметров из потока
StreamHelperArg	Класс который используется для передачи параметров между объектами класса StreamHelper
abstract	Абстрактного класса, который сохраняет/загружает данные коллекции в файл
Timer	Интерфейс для работы с SimpleTimer
SimpleTimer	Класс SimpleTimer
Wheel	
CarWheel	Класс описывающий колесо машины

Диаграмма классов

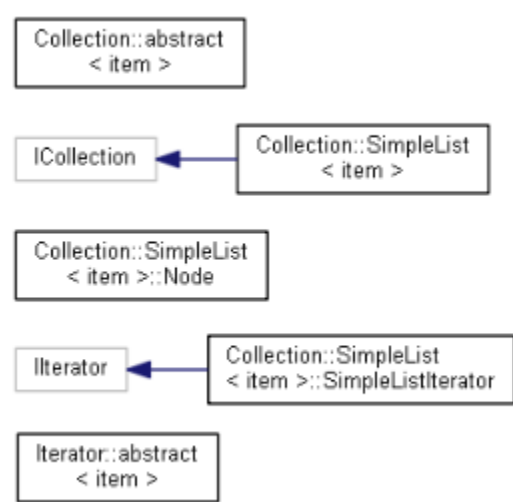


Рисунок 1 -Диаграмма классов контейнеров

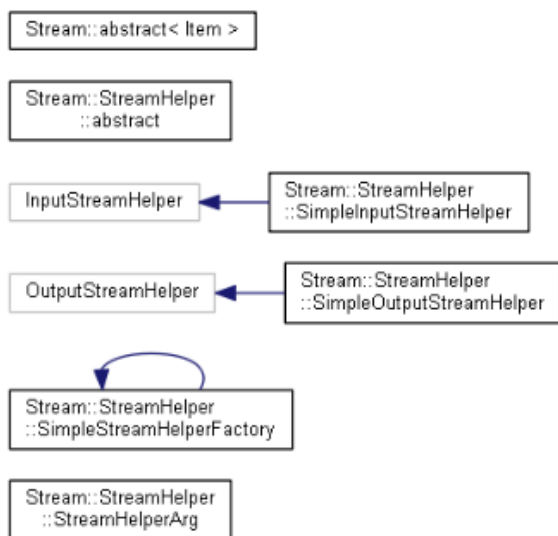


Рисунок 2 -Диаграмма классов для сохранения данных из контейнеров

Разработанные методы и функции

См. раздел **members** на заголовочных страницах соответствующих типов данных

Ссылки на файлы проекта

GraphScreenCreator.hpp	Содержит объявление класса GraphScreenCreator
ICollection.h	Содержит объявление класса ICollection
Iiterator.h	Содержит объявление класса Iiterator
IOCollection.cpp	Содержит реализацию абстрактного класса IOCollection
IOCollection.h	Содержит объявление абстрактного класса IOCollection
Main.cpp	
OutputStreamHelper.h	Содержит объявление абстрактного класса OutputStreamHelper
SimpleList.cpp	Реализация класса SimpleList
SimpleList.h	Содержит объявление класса SimpleList
SimpleListIterator.cpp	

Текст программы

```
Wheel::CarWheel* StringToObject1(std::string type) {
    if (type == std::string("CarWheel"))return new Wheel::CarWheel();
    else return NULL;
}
Wheel::Wheel* StringToObject2(std::string type) {
    if (type == std::string("Wheel"))return new Wheel::Wheel();
    if (type == std::string("CarWheel"))return new Wheel::CarWheel();
    else return NULL;
}
int main() {
    {
        Collection::ICollection<Wheel::CarWheel*> *carWheelList=NULL; // контейнер, который будет хранить указатели
        auto sh = new Stream::StreamHelper::SimpleStreamHelperFactory();
        carWheelList = Stream::IOCollection::Load<Wheel::CarWheel*>("wheelList",
            "CarWheelListLoad.txt", sh, StringToObject1); // загружаем данные из файла
        DemonstratingDeletion(carWheelList);
        auto it1 = carWheelList->CreateIterator(); // получение итератора
        CarWheelListShow(it1); // выводим на экран
        DemonstratingAddition(carWheelList);
        Stream::IOCollection::Save<Wheel::CarWheel*>(carWheelList,"wheelList", "CarWheelListSave.txt", sh);
        CarWheelListShow(it1); // выводим на экран
        Collection::ICollection<Wheel::Wheel*> *wheelList = NULL; // контейнер, который будет хранить указатели
        wheelList = Stream::IOCollection::Load<Wheel::Wheel*>("wheelList", "CarWheelListSave.txt", sh, StringToObject2);
        wheelList->Push(new Wheel::Wheel(42, 60, EUNITS_CENTIMETERS)); // добавим еще один элемент
        Stream::IOCollection::Save<Wheel::Wheel*>(wheelList, "wheelList", "WheelListSave.txt", sh); // сохраняем
        auto it2 = wheelList->CreateIterator(); // получение
        WheelListShow(it2); // выводим на экран
    }
    Очистка памяти
    _CrtDumpMemoryLeaks();
    return 0;
}

void CarWheelListShow(Collection::ICollection<Wheel::CarWheel*>::iterator it) {
    for (it->First(); !it->IsDone(); it->Next()) {
        auto sc = new Screen::ScreenCreator::CarWheelScreenCreator(it->CurrentItem());
        Show(sc);
        delete sc;
    }
}

void WheelListShow(Collection::ICollection<Wheel::Wheel*>::iterator it) {
    for (it->First(); !it->IsDone(); it->Next()) {
        auto sc = new Screen::ScreenCreator::DefaultScreenCreator(it->CurrentItem());
        Show(sc);
        delete sc;
    }
}

void DemonstratingDeletion(Collection::ICollection<Wheel::CarWheel*> *wheelList) {
    auto forRemove = (*wheelList)[1]; // получим второй элемент "MP-16 ", "Matador"
    wheelList->Remove(forRemove); // удалим второй элемент из коллекции
    delete forRemove; // освободим память
    forRemove = (*wheelList)[0]; // получим первый элемент "UltraGrip Performance G1", "Goodyear"
    wheelList->RemoveAt(0); // удалим первый элемент из коллекции
    delete forRemove; // освободим память
    forRemove = wheelList->Pop(); // достанем элемент из конца коллекции "Hakkapeliitta 9 (шип)", "Nokian"
    delete forRemove; // освободим память
}

void DemonstratingAddition(Collection::ICollection<Wheel::CarWheel*> *wheelList) {
    wheelList->Insert(0, new Wheel::CarWheel(514.5, 295, EUNITS_CENTIMETERS,
        "Proxes T1 Sport SUV 295/40", "Тою")); // вставим элемент во вторую позицию
    wheelList->Push(new Wheel::CarWheel(30, 50, EUNITS_CENTIMETERS, "UltraGrip Performance G1", "Goodyear"));
    wheelList->Push(new Wheel::CarWheel(508, 500, EUNITS::EUNITS_MILLIMETRES, "MP-16 ", "Nokian"));
    wheelList->Push(new Wheel::CarWheel(508, 275, EUNITS::EUNITS_MILLIMETRES, "Hakkapeliitta 9 (шип)", "Nokian"));
}
```

Результаты работы

```
1 wheelList : {
2   size : 4
3   collection : [
4     {
5       BEGIN : CarWheel
6       tireManufacturer : Goodyear
7       tireName : UltraGrip Performance G.
8       BEGIN : Wheel
9       diameter : 30
10      units : 0
11      width : 50
12    },
13    {
14      BEGIN : CarWheel
15      tireManufacturer : Matador
16      tireName : MP-16
17      BEGIN : Wheel
18      diameter : 304.8
19      units : 1
20      width : 500
21    },
22    {
23      BEGIN : CarWheel
24      tireManufacturer : Continental
25      tireName : WinterContact TS 860
26      BEGIN : Wheel
27      diameter : 355.6
28      units : 1
29      width : 185
30    },
31    {
40  ]}
```

Рисунок 1 - Загружаемые данные из файла в коллекцию типа CarWheel

```
-----
Diameter: 355.6
Width: 185
Units: MILLIMETRES
Volume: 1.83732e+07
Tire Manufacturer: Continental
Tire Name: WinterContact TS 860
-----
```

Рисунок 2 - Результат демонстрации удаления

```

1 wheelList : {
2   size : 5
3   collection : [
4     {
5       BEGIN : CarWheel
6       tireManufacturer : Continental
7       tireName : WinterContact TS 860
8       BEGIN : Wheel
9       diameter : 355.6
10      units : 1
11      width : 185
12    },
13    {
14      BEGIN : CarWheel
15      tireManufacturer : Toyo
16      tireName : Proxes Tl Sport SUV 295/40
17      BEGIN : Wheel
18      diameter : 514.5
19      units : 0
20      width : 295
21    },
22    {
23      BEGIN : CarWheel
24      tireManufacturer : Goodyear
25      tireName : UltraGrip Performance G1
26      BEGIN : Wheel
27      diameter : 30
28      units : 0
29      width : 50
30    },
31  ],
40  {
49  }}

```

Рисунок 3 - Результат демонстрации добавления с выводом данных в файл

```

-----
Diameter: 355.6
Width: 185
Units: MILLIMETRES
Volume: 1.83732e+07
-----
Diameter: 514.5
Width: 295
Units: CENTIMETERS
Volume: 6.13313e+07
-----
Diameter: 30
Width: 50
Units: CENTIMETERS
Volume: 35342.9
-----
Diameter: 508
Width: 500
Units: MILLIMETRES
Volume: 1.01341e+08
-----
Diameter: 508
Width: 275
Units: MILLIMETRES
Volume: 5.57378e+07
-----

```

Рисунок 4 - Загружаемые данные из файла в коллекцию типа Wheel

```

1  wheelList : {
2      size : 6
3      collection : [
4          {
5              BEGIN : CarWheel
6              tireManufacturer : Continental
7              tireName : WinterContact TS 860
8              BEGIN : Wheel
9              diameter : 355.6
10             units : 1
11             width : 185
12         },
13         {
22         {
31         {
40         {
41             BEGIN : CarWheel
42             tireManufacturer : Nokian
43             tireName : Hakkapeliitta 9 (мин)
44             BEGIN : Wheel
45             diameter : 508
46             units : 1
47             width : 275
48         },
49         {
50             BEGIN : Wheel
51             diameter : 42
52             units : 0
53             width : 60
54         }
55     ]}

```




Рисунок 5 - Результат добавления одного элемента типа Wheel и выводов всей коллекции в файла

```

"shazhko06.exe" (Win32). Загружено "D:\GitHub\kit25a\ashazhko-artem\src\x64\Debug\shazhko06.exe". Символы загружены.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\ntdll.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\kernel32.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\KernelBase.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\vcruntime140d.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\ucrtbased.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\msvcpr140d.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\kernel.appcore.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\msvcrt.dll". Невозможно найти или открыть PDB-файл.
"shazhko06.exe" (Win32). Загружено "C:\Windows\System32\rpcrt4.dll". Невозможно найти или открыть PDB-файл.
Поток 0x24d4 завершился с кодом 0 (0x0).
Поток 0x7d8 завершился с кодом 0 (0x0).
Поток 0x1a4c завершился с кодом 0 (0x0).
Программа "[5504] shazhko06.exe" завершилась с кодом 0 (0x0).

```

Рисунок 5 - Утечки памяти нет

```

[=====] Running 6 tests from 6 test cases.
[-----] Global test environment set-up.
[-----] 1 test from SimpleList_save
[ RUN    ] SimpleList_save.saveCollection
[      OK ] SimpleList_save.saveCollection (24 ms)
[-----] 1 test from SimpleList_save (25 ms total)

[-----] 1 test from SimpleList_load
[ RUN    ] SimpleList_load.loadCollection
[      OK ] SimpleList_load.loadCollection (24 ms)
[-----] 1 test from SimpleList_load (25 ms total)

[-----] 1 test from SimpleList_removeByIndex
[ RUN    ] SimpleList_removeByIndex.oneItemCollection
[      OK ] SimpleList_removeByIndex.oneItemCollection (1 ms)
[-----] 1 test from SimpleList_removeByIndex (1 ms total)

[-----] 1 test from SimpleList_removeByItem
[ RUN    ] SimpleList_removeByItem.oneItemCollection
[      OK ] SimpleList_removeByItem.oneItemCollection (1 ms)
[-----] 1 test from SimpleList_removeByItem (1 ms total)

[-----] 1 test from SimpleList_Insert
[ RUN    ] SimpleList_Insert newItemAtSecondPositionCollection
[      OK ] SimpleList_Insert newItemAtSecondPositionCollection (1 ms)
[-----] 1 test from SimpleList_Insert (3 ms total)

[-----] 1 test from SimpleList_push
[ RUN    ] SimpleList_push.twoItemsInCollection
[      OK ] SimpleList_push.twoItemsInCollection (0 ms)
[-----] 1 test from SimpleList_push (1 ms total)

[-----] Global test environment tear-down
[=====] 6 tests from 6 test cases ran. (67 ms total)
[ PASSED ] 6 tests.

```

Рисунок 5 - Выполнение тестов

Выводы

В ходе лабораторной работы были получены практические навыки создания собственных контейнеров для объектов хранения ранее разработанных классов