Лабораторная работа №1

по курсу
«Информатика (организация и поиск данных)»
(3 семестр)

Варианты заданий

Постановка задачи

Реализовать идиому «умный указатель» для обеспечения автоматического управления памятью, показать эффективность полученной реализации (отсутствие утечек). Использование стандартных реализаций (например, STL) запрещено, кроме сравнительного тестирования. Относящиеся к реализации умных указателей классы должны быть шаблонными, а также должна корректно поддерживаться подтипизация шаблонных аргументов (см. ниже).

Минимальные требования к программе. Концепция умных указателей может быть реализована по-разному, но в любом случае реализация должна быть снабжена функциональными и нагрузочными тестами. Последние должны показывать затраты по времени и по памяти на обслуживание умных указателей и сравнение с вариантом без них (опционально – еще сравнение с реализацией STL). При оценке количественных показателей следует ориентироваться на малое число аллокацируемых объектов (от нескольких штук до нескольких тысяч), так и на большое: порядка $10^6 - 10^8$. Результаты всех измерений следует представлять как в табличном, так и в графическом виде. Пользовательский интерфейс может быть как консольным, так и графическим.

Методические указания. Предлагается 4 основных варианта выполнения работы:

- **централизованное хранение объектов**: предполагает реализацию класса SmrtPtr<T>, отвечающего за подсчет ссылок на указываемый объект и удаление этого объекта, когда число ссылок достигает 0;
- **децентрализованное хранение объектов**: реализуется два шаблонных класса – UnqPtr<T> («ведущий» указатель, master pointer, аналог unique_ptr из STL) и ShrdPtr<T> («дескриптор», handler, аналог shared_ptr из STL)
- **децентрализованное хранение объектов (вариант 2)**: реализуется два шаблонных класса – UnqPtr<T> («ведущий» указатель, master pointer, аналог unique_ptr из STL) и ShrdPtr<T> («дескриптор», handler, аналог shared_ptr из STL), однако подсчет ссылок ведется только средствами ShrdPtr, за счет поля size_t *referenceCount. Засчет использования указателя, удается подсчитывать количество ссылок без создания дополнительных объектов.
- **с арифметикой указателей**: объекты хранятся централизовано в некотором контейнере и на них можно получиться специальный указатель, для которого безопасной реализована арифметика указателей (классы MemorySpan<T> и MsPtr<T>).

Краткая характеристика ключевых АТД:

Класс	Назначение	Особенности	
`SmrtPtr<T>`	Упрощенная реализация, решающая основную задачу –	Указываемые объекты никогда не копируются,	

	своевременное освобождение более не нужной памяти.	хранятся в некотором скрытом от пользователя контейнере. Ведется подсчет ссылок на каждый объект; объект удаляется, как только число ссылок на него становится 0.	
`UnqPtr<T>`	Отвечает за хранение одного экземпляра типа T. Помогает реализовать семантику передачи параметров по значению.	Объекты только копируются, нельзя, чтобы несколько UnqPtr указывали на один и тот же хранимый объект.	
`ShrdPtr<T>`	Реализует семантику передачи параметров по ссылке.	Непосредственно может ссылаться только на UnqPtr, прямой ссылки на хранимый объект не допускается.[1]	
`MemorySpan<T>`	Коллекция типа «массив». Доступ к объектам – только через специальные классы умных указателей (SmrtPtr, UnqPtr, ShrdPtr, MsPtr)	Упрощенно, это модификация ArraySequence, в которой вместо T Get(size_t) реализованы: – ShrdPtr<T> Copy(size_t) – UnqPtr<T> Get(size_t) – MsPtr<T> Locate(size_t)	
`MsPtr<T>`	Специализированный вариант умного указателя, предназначенный только для использования совместно с MemorySpan и реализующий арифметику указателей.	За счет того, что указатель привязан к конкретному MemorySpan, никогда не может выйти за границы массива.	

Для проверки корректности реализации умных указателей следует реализовать с их помощью какой-либо контейнер. Рекомендуется это сделать для системы Sequence (DynamicArray и LinkedList – не обязательно).

Следует учесть наличие подтипизации на типах в C++. В частности, если $C_1 \subseteq C_2$, то:

```
SmrtPtr<C1> ptr1 = …; // каким-либо образом создаем умный указатель
SmrtPtr<C2> ptr2 = ptr1; // должно работать
```

При этом если C_1 не является подклассом C_2, то присваивание (и другие аналогичные операции) не должно срабатывать. Достичь этого можно с помощью `static_cast`/`dynamic_cast` или концептов (за счет наложения ограничений на второй типовый параметр оператора = или/и копирующего конструктора).

[1] В принципе, в рамках децентрализованной схемы (вариант 2) можно напрямую указывать на управляемый объект. Однако это означает дублирование логики контроля и освобождения ресурсов. Хотя логика, в большинстве случаев, несложная.

Критерии оценки

1.	Качество программного кода:	– стиль (в т.ч.: имена, отступы и проч.) (0-2) – структурированность (напр. декомпозиция сложных функций на более простые) (0-2) – качество основных и второстепенных алгоритмов (напр. обработка граничных случаев и некорректных исходных данных и т.п.) (0-3)	0-7 баллов
2.	Объем реализации:	– SmrtPtr (0-1) – Хранилище для SmrtPtr (0-2) – UnqPtr (0-1,5) – ShrdPtr (0-1,5) – MemorySpan (0-2) – MsPtr (0-1)	0-3 (бонус 0-6) баллов
3.	Качество тестов	– степень покрытия – читаемость – качество проверки (граничные и некорректные значения, и др.)	0-2 баллов
4.	Владение теорией	знание алгоритмов, области их применимости, умение сравнивать с аналогами, оценить сложность, корректность реализации	0-3 баллов
		Итого	0-15 (бонус 0-10) баллов