НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ
«МИФИ»

Институт ЛаПлаз

Кафедра №31 «Прикладная математика»

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

«Объектно-ориентированное программирование»

Задание: Класс вектора

Требуется реализовать класс `Vector`, который должен поддерживать два режима управления памятью: владеющий и невладеющий. Во владеющем режиме объект самостоятельно выделяет память под элементы вектора и отвечает за её освобождение. В невладеющем режиме объект работает с внешним буфером данных, предоставленным извне, и не управляет его жизненным циклом. Для отслеживания режима работы класс должен хранить специальный флаг владения. Также необходимо хранить указатель на массив элементов и размер вектора.

Класс должен поддерживать следующие конструкторы:

- Конструктор по умолчанию, создающий пустой вектор (владеющий режим);
- Конструктор с параметром размера, создающий вектор заданного размера с нулевыми элементами (владеющий режим);
- Конструктор, принимающий размер и значение, создающий вектор заданного размера, инициализированный указанным значением (владеющий режим);
- Конструктор, принимающий указатель на внешний массив и его размер (невладеющий режим);
- Конструктор копирования, сохраняющий режим владения: при копировании владеющего вектора создается новый владеющий вектор, при копировании невладеющего — новый невладеющий вектор.

Класс должен предоставлять методы:

- `Size()` — получение размера вектора;
- `OwnsMemory()` — проверка режима владения памятью;
- `SetElement(index, value)` — установка значения элемента;
- `GetElement(index)` — получение значения элемента;
- `Dot(other)` — скалярное произведение с другим вектором (метод должен быть эффективным, не создавая ненужных копий данных при вычислении);
- `Link(external_ptr, size)` — переключение в невладеющий режим с привязкой к внешнему буферу;
- `Unlink()` — переключение во владеющий режим.

Метод `Link` позволяет переиспользовать объект для работы с разными внешними буферами. Метод `Unlink` обеспечивает независимость объекта от внешних данных, создавая собственную копию.

Необходимо продемонстрировать полноценную работу реализованного класса, показав все возможные сценарии использования. Демонстрация должна охватывать различные способы создания векторов, их взаимодействие между собой, переходы между режимами владения памятью и корректность работы всех методов в различных ситуациях. Также необходимо показать, что реализация корректно обрабатывает исключительные ситуации и некорректные входные данные.

Реализация должна демонстрировать понимание принципов работы с динамической памятью. Использование контейнеров и умных указателей из стандартной библиотеки запрещено. Вся работа с памятью должна выполняться через операторы `new` и `delete`.