**РАБОТА No6 ПРАКТИЧЕСКАЯ. ОБРАТНОЕ**

**ПРОЕКТИРОВАНИЕ АЛГОРИТМА**

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является получение практических выполнения обратного проектирования.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЯ ДЛЯ РАБОТЫ

Заданием для работы является исходный код, содержащий описание классов. Классы должны содержать операции и их реализацию.

Выполнение должно включать следующие этапы.

1. Анализ кода и выделение состав классов.

**Переменные:**

char \*string; //Указатель структуры

int buffer[100]; //Временный буфер

char name[10]={«Massiv»}; // Имя данных

int a; // Целое число

int b; // Целочисленная пустая строка (пустая); // конструктор

int tmp\_a; //Целочисленная переменная

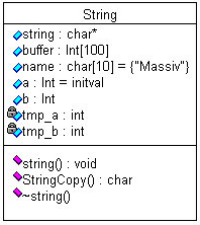
int tmp\_b;// Целочисленная переменная

**Методы**:

void ~string(void); // деструктор

char StringCopy(char \*, //Buffer char \*, //sourcel char \*); // метод копирования строки

1. Отображение классов в среде моделирования. Для каждого класса в отчёте по работе должно быть сделано пояснение – на основе какого программного объекта он выявлен.



Тут он выявлен способом обратного проектирования

1. Выявление переменных классов и отображение их в среде моделирования. Для каждой переменной класса в отчёте по лабораторной работе должно быть сделано пояснение.
2. Класс, отвечающий за хранение и обработку строки:

Программные объекты: переменная string и функции string(), StringCopy().

Этот класс содержит методы для работы со строками,

такие как получение длины строки и т.д.

1. Класс, отвечающий за хранение и обработку целочисленных значений:

Программные объекты: переменные a, b, tmp\_a и tmp\_b.

Этот класс может содержать методы для работы с

целочисленными значениями, такие как сравнение чисел, выполнение

арифметических операций и т.д.

1. Класс, отвечающий за управление временным буфером:

Программные объекты: переменная buffer.

Этот класс может содержать методы для работы с буфером,

такие как запись и чтение данных, расчет количества доступной памяти,

проверку наличия свободного места, и т.д.

1. Класс, отвечающий за инициализацию и уничтожение объектов:

Программные объекты: функции string() , ~string().

Пояснение: данный класс может содержать методы для инициализации и

удаления объектов, а также методы для управления ресурсами,

выделенными для работы с объектами (например, освобождение памяти).

1. Класс, отвечающий за хранение имени данных:

Программные объекты: переменная name.

Этот класс может содержать методы для работы с именами

данных, такие как проверка наличия имени, получение имени, установка

имени, и т.д.

1. Класс, отвечающий за выполнение операций с данными:

Программные объекты: переменные string, a, b и buffer, функции string(),

~string() и StringCopy().

Этот класс может содержать методы для выполнения

операций с данными, такие как копирование данных из буфера в строку,

сортировка целочисленных значений, преобразование типов данных, и т.д.

1. Выявление операций классов. Для каждого класса необходимо выявить операции и сделано пояснение о свойствах операции, её параметрах и уровне в иерархии наследования.
2. void string(void) - это конструктор класса.

Он используется для инициализации переменных объекта класса при его

создании. Эта операция относится к конструктору класса string. Свойства

операции: отсутствуют параметры.

1. void ~string(void) - это деструктор класса. Он вызывается автоматически,

когда объект класса удаляется из памяти. Он используется для освобождения

ресурсов, которые были выделены объекту во время его существования. Эта

операция относится к деструктору класса string. Свойства операции:

отсутствуют параметры.

1. char StringCopy(char \*, char \*, char \*) - это метод класса string. Он

используется для копирования данных из одной строки в другую. Свойства

операции: параметры - указатель на буфер, из которого будет скопирована

строка, указатель на буфер, в который будет скопирована строка, и указатель

на вторую строку, которая будет скопирована. Уровень в иерархии

наследования - метод класса string.

1. Переменные string, buffer и name могут быть в классе string.
2. a и b - это переменные типа int, которые могут использоваться в методах

класса string или в других классах, если они есть.

1. tmp\_a и tmp\_b - это переменные типа int, которые, вероятно, используются

внутри методов класса string.

**Вывод**: Я получила практические навыки выполнения обратного проектирования.

**4. КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что из себя представляет обратное проектирование?

Обратное проектирование - это специальный метод, который позволяет воссоздать объект в виде виртуальной модели по существующей физической детали. Полученная 3D модель в дальнейшем может использоваться для конструирования (CAD), производства (CAM) и проектирования на компьютере (CAE). Упрощенно этот процесс можно представить как измерение объекта специальным прибором (3D сканером) и последующую реконструкцию в виде трехмерной модели.

1. Для чего выполняется обратное проектирование?

Обратное проектирование используется для анализа функциональности продукта, его компонентов, оценки стоимости и определения возможных случаев нарушения патентных прав.

1. Что из себя представляет результат обратного проектирования?

Обратным проектированием (Reverse engineering) называется

процесс преобразования в модель кода, записанного на каком-либо

языке программирования.

Результат обратного проектирования - параметрическая модель

1. Какие модели можно построить в результате обратного проектирования?

В результате этого процесса вы получаете огромный объем информации, часть которой находится на более низком уровне детализации, чем необходимо для построения полезных моделей. В то же время обратное проектирование никогда не бывает полным. Как уже упоминалось, прямое проектирование ведет к потере информации, так что полностью восстановить модель на основе кода не удастся, если только инструментальные средства не включали в комментариях к исходному тексту информацию, выходящую за пределы семантики языка реализации.

1. Существуют ли инструментальные средства для обратного проектирования?

Инструментальные средства для разработки баз данных существуют достаточно давно, но их применение в практическом проектировании началось совсем недавно (примерно в начале XXI в.), но также не всегда в полном объеме. Это привело к тому, что сегодня много баз данных работает без наличия соответствующих им моделей баз данных, что вызывает серьезные сложности в их сопровождении и модификации. С учетом этой проблемы в инструментальных средствах реализуется возможность восстановления структуры базы данных (модели) из физической базы данных, что называется обратным проектированием (Reverse Engineering). Принцип работы данного инструмента в IBM InfoSphere Data Architect идентичен процедуре перехода от модели одного уровня к модели другого уровня.