**Задача 3: перегрузка операторов**

Вариант 30

1. Разработать класс "двоичный вектор" в соответствии со следующим заданием:

Состояние класса -

Вектор включает в себя элементы типа char, значения которых только 0 и 1, и определяется размером (количество элементов в векторе) и значениями своих элементов, например: вектор из 6 элементов {1, 0, 1, 1, 0, 0}. Память под вектор выделяется статически, во время компиляции, и задается массивом фиксированного предельного значения (например, максимальный размер вектора 100 элементов). При выполнении операций с векторами разной длины расширение векторов нулями выполнять слева.

Протокол класса -

Определяет возможности создания и инициализации экземпляров класса и правила использования их (методы класса).

Предусмотреть следующие возможности:

• пустой конструктор для инициализации экземпляров и массивов экземпляров класса по умолчанию;

• создание экземпляров класса с инициализацией значением типа int;

• создание экземпляров класса с инициализацией значениями элементов вектора как строки символов;

• ввод экземпляров класса из входного потока и вывод их значений в выходной поток (с помощью перегруженных операторов >> и << );

• выполнение поразрядной логической операций: включающее ИЛИ двух векторов (с помощью перегруженного оператора | );

• выполнение поразрядной логической операций: И двух векторов (с помощью перегруженного оператора & );

• выполнение поразрядной логической операции исключающее ИЛИ (с помощью перегруженного оператора ^= ),

• получение “дополнения” (с помощью перегруженного оператора ~);

• выделение вектора ограниченного слева и справа единицами, исключив левые и правые нули (например, для {0, 0, 1, 0, 1, 1, 0} получить {1, 0, 1, 1} ).

2. Проектирование класса рекомендуется начать с представления состояния класса, учитываю-щего заданные операции, а затем реализации конструкторов и перегруженного оператора вывода. Для отладки и исчерпывающего тестирования других методов разработанного класса реализовать диалоговую программу, которая позволяет вводить параметры, отлаживаемых методов. Для обра-ботки ошибочных ситуаций использовать механизм исключительных ситуаций.

3. Повторить разработку класса при условии, что память под вектор необходимой длины выде-ляется динамически, во время выполнения программы (с помощью оператора new; память задается указателем на char в состоянии класса).

Дополнить интерфейс класса следующими возможностями:

• создание экземпляра класса с его инициализацией другим экземпляром класса (копирую-щий конструктор);

• переопределение экземпляра класса (с помощью перегруженного оператора присваива-ния).

 4. Написать прикладную программу, использующую разработанный класс.