Лабораторная работа №2

по программированию на языке С++

“Классы на C++ Builder”

**Задание:**

Составить на языке C++ описания классов для указанных объектов. Написать методы класса, демонстрирующие использование объектов разработанного класса.

Множество целых чисел на основе массива(включение,исключение,

Класс, применяемый в программе:

class Set1

{

private:

int chislo;

public:

int size1;

int \*mas;

Set1() : size1(0),mas(new int[0]) {};

~Set1();

void Add(int elem); // включение элемента

void Del(int chisl); // исключение элемента

int Poisk(int chisl); // поиск элемента

friend Set1 operator&(Set1& A,Set1& B); // Объединение

friend Set1 operator|(Set1& A,Set1& B); // Пересечение

};

**Описание работы программы:**

При запуске программы пользователь может выбрать действия, которые будут произведены над первым или вторым множеством. Он может включить/исключить элементы в первое/второе множество, найти элемент в первом/втором множестве, а также выполнить объединение или пересечение множеств. Каждая из этих функций вызывается после нажатия пользователем на одну из кнопок, размещённых на форме.

Функция добавления сначала проверяет размер множества. Если он равен нулю, то она просто добавляет введенный пользователем элемент в множество, иначе она пробегается по всем элементам множества, сверяя его с введённым. Если в множестве находится такой же элемент то множество остаётся без изменения, иначе элемент добавляется и размер множества size1 увеличивается на 1.

Функция поиска получает от пользователя элемент который необходимо найти и в цикле пробегает по всем элементам множества, сравнивая его с полученным. Если элемент нашёлся, то запоминается его индекс и выводится текстовое сообщение, что элемент найден под некоторым индексом. Если элемент не был найден, то выводится текстовое сообщение, о том, что такого элемента нет во множестве.

Функция удаления таким же образом, как и в функция поиска находит нужный элемент во множестве(если он есть), запоминает его индекс и начиная с элемента множества, стоящего под данным индексом сдвигает все последующие элементы во множестве на 1 влево. После чего размер множества size1 уменьшается на 1.

Функция объединения принимает в качестве параметров два экземпляра класса Set1 A и Set1 B, являющиеся соответственно первым и вторым множеством. Создаётся новый экземпляр класса Set1 P, куда сначала с помощью вызова функции добавления void Set1::Add(int elem) добавляются все элементы из первого множества. Далее проверяются все элементы из второго множества на предмет вхождения в первое. Если элемент не входит, то вызывается функция void Set1::Add(int elem) для этого элемента.

Функция пересечения так же как и функция объединения принимает в качестве параметров два экземпляра класса и создает новый экземпляр Set1 C. Проверяются все элементы второго множества. Если элемент из этого множества входит в первое, то запоминается индекс данного элемента и для него вызывается функция void Set1::Add(int elem). Таким образом в Set1 C оказываются только элементы входящие одновременно в оба множества.

**Алгоритмы выполнения основных операций на псевдокоде:**

**Функция добавления:**

{

Если(размер множества == 0)

{

Размер множества += 1

Добавляем элемент в множество

}

Иначе

{

Пока(k < размер множества)

{

Если (введённый элемент == элементу множества)

Выход

k += 1

}

Если(k >= размера множества)

{

Размер множества += 1

Добавление элемента в множество

}

}

}

**Функция поиска:**

**{**

Пока(k < размер множества)

{

Если (введённый элемент == элементу множества)

Выход

k += 1

}

Если (введённый элемент == элементу множества)

{

Запоминаем индекс элемента

}

Return индекс элемента

}

**Функция удаления:**

**{**

Пока(k < размер множества)

{

Если (введённый элемент == элементу множества)

Выход

k += 1

}

Если (введённый элемент == элементу множества)

{

Запоминаем индекс элемента

Начиная с элемента, стоящего под данным индексом сдвигаем элементы влево на 1

Размер множества -= 1

}

**Функция объединения:**

{

Новый экземпляр класса P

Пока(не конец 1-го множества)

{

Добавляем элемент в P

}

Пока(не конец 2-го множества)

{

Пока(не конец 1-го множества)

{

Если(элемент из 1-го множества == элемент из 2-го множества)

Выход

}

Если(одинаковых элементов не найдено)

{

Добавление элемента в P

}

}

Return P;

}

**Функция пересечения:**

{

Новый экземпляр класса C

Пока(не конец 2-го множества)

{

Пока(не конец 1-го множества)

{

Если(элемент из 1-го множества == элемент из 2-го множества)

Выход

}

Если(элемент присутствует в обоих множествах)

{

Добавление элемента в C

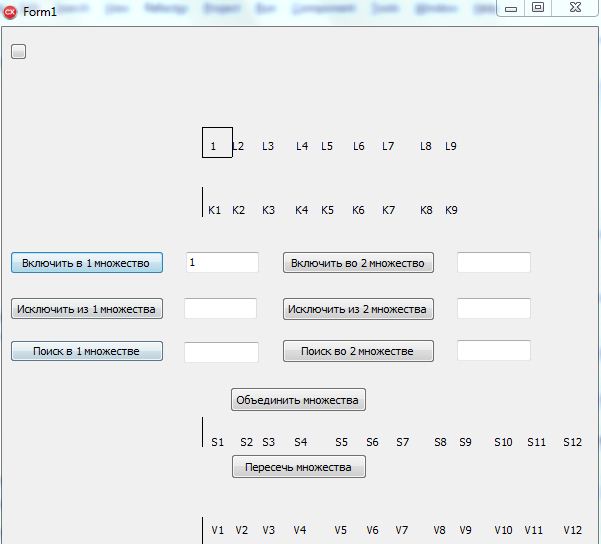
}

}

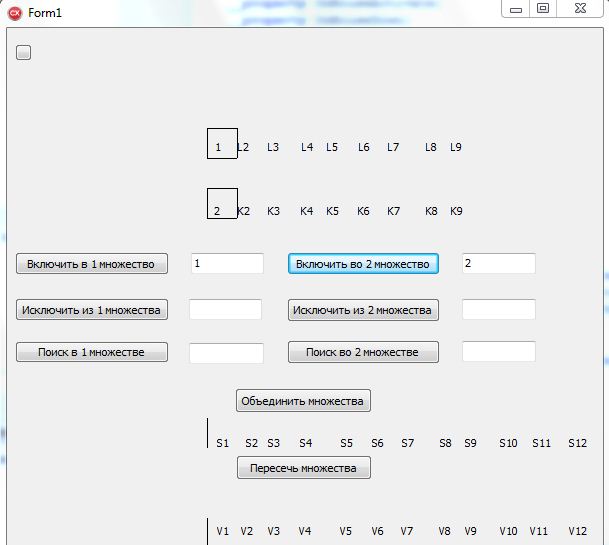
Return C;

}

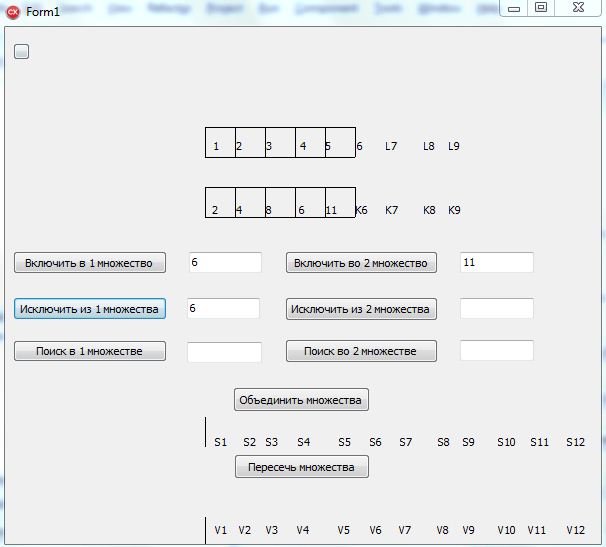
**Распечатки экранов при работе программы:**

Добавление в первое множество:

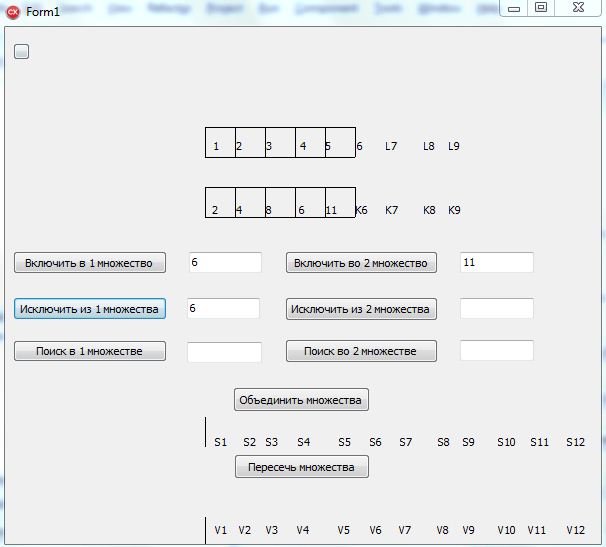
Добавление во второе множество:



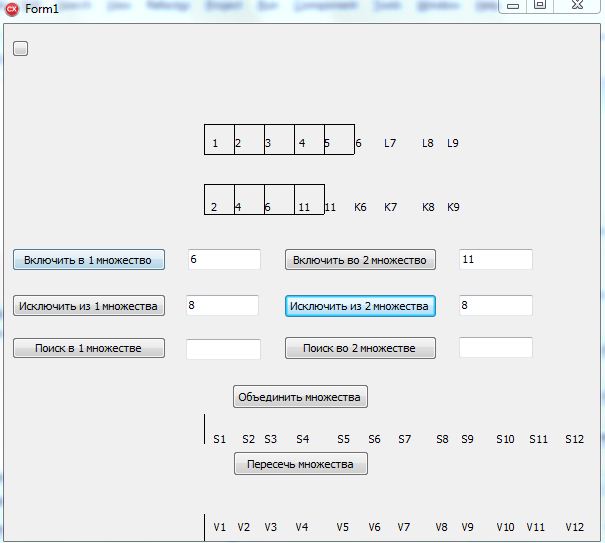
Таким же образом добавим остальные элементы в первое и второе множество:



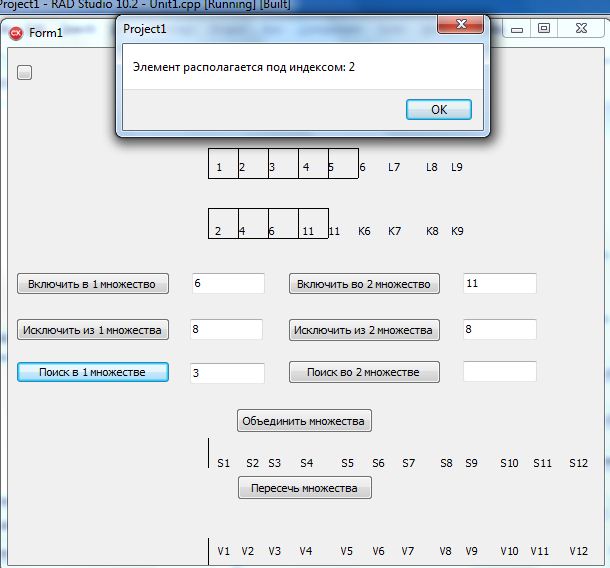
Исключим элемент из 1-го множества:

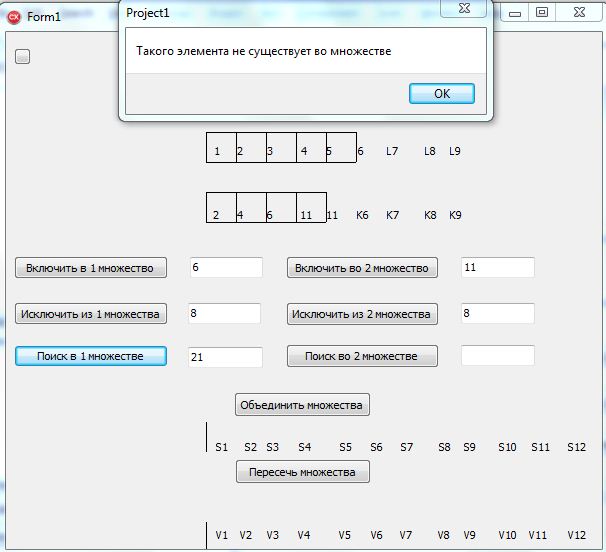


Исключим элемент из 2-го множества:

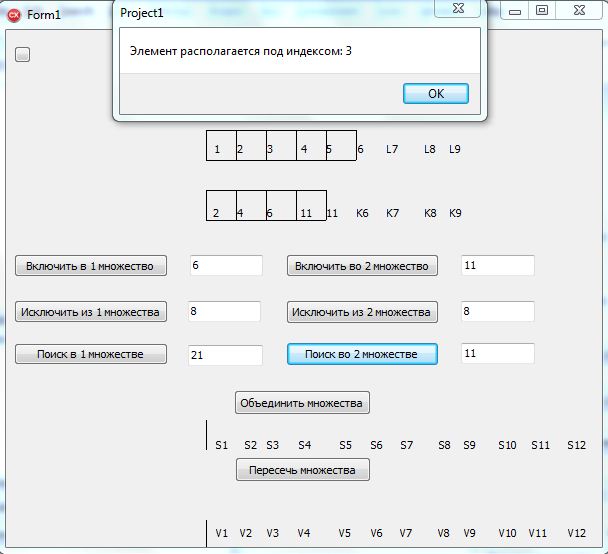


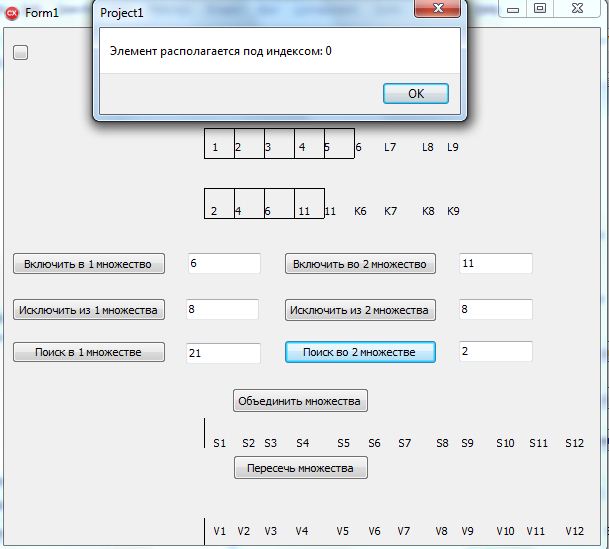
Найдём элемент в 1-ом множестве:



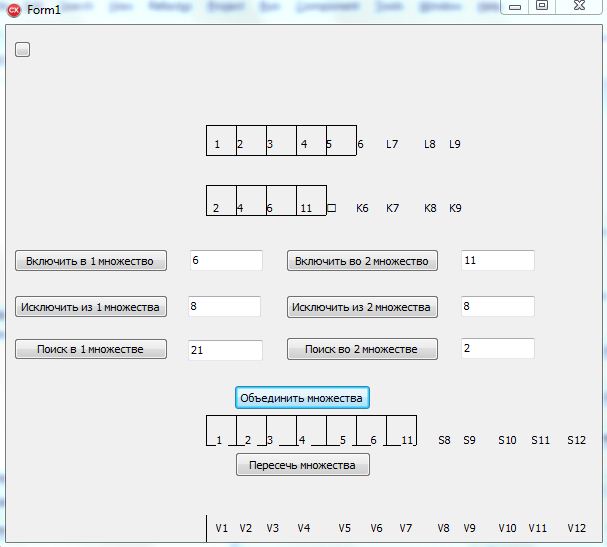


Найдём элемент в 2-ом множестве:





Объединение множеств:



Пересечение множеств:

