Автономная некоммерческая организация
дополнительного профессионального образования
(АНО ДПО «Инфосфера»)
Центр профессиональной подготовки
ИНСТИТУТ ПРОГРАММНЫХ СИСТЕМ

Рабочая программа дисциплины
«Объектно-ориентированное программирование»

Разработал:
преподаватель ИПС
АНО ДПО «Инфосфера»
А.Н. Малов

Йошкар-Ола, 2017

Пояснительная записка

Целью курса является получение знаний об основных концепциях объектно-ориентированного программирования и его применения для разработки и проектирования программных систем. В результате прохождения данного курса студент получает навыки объектно-ориентированного проектирования и анализа при разработке программ, умение использовать наследование и полиморфизм, применять на практике основные паттерны проектирования.

Планируемые результаты обучения

Осуществляется предварительная самостоятельная или под руководством разработка алгоритмов с использование графических средств (блок-схемы, UML-диаграммы и др.). Не требуется взаимодействие с другими программистами, системным аналитиком и архитектором программного обеспечения. Осуществляется решение типовых задач. Полученные результаты представляются руководителю разработки программного обеспечения.

Осуществляется самостоятельное или под руководством оформление программного кода в соответствии с внутренними нормативными документами организации (регламентами, приказами, порядками) и, при необходимости, ГОСТами. Не требуется взаимодействие с другими программистами, системным аналитиком и архитектором программного обеспечения. Осуществляется решение типовых задач. Полученные результаты представляются руководителю разработки программного обеспечения.

Регистрация новых версий программного обеспечения осуществляется с использованием системы контроля версий, принятой в организации и в соответствии с утвержденными внутренними нормативными документами организации. Не требуется взаимодействие с другими программистами, системным аналитиком и системным архитектором. Осуществляется решение типовых задач. Полученные результаты представляются руководителю разработки программного обеспечения.

Проверка работоспособности программного кода осуществляется на основании функциональных требований и технических спецификаций на программное обеспечение. Не требуется взаимодействие с другими программистами, системным аналитиком и архитектором программного обеспечения. Осуществляется решение типовых задач. Полученные результаты представляются руководителю разработки программного обеспечения.

Разработка тестовых наборов данных предполагает формулирование правил создания, структуры и требований к тестовым наборам данных, подготовка наборов данных, используемых в процессе проверки работоспособности. Решаются задачи с элементами проектирования. Не требуется

взаимодействие с другими программистами, системным аналитиком и архитектором программного обеспечения. Полученные результаты представляются руководителю разработки программного обеспечения.

Проверка работоспособности программного обеспечения осуществляется программистом на основании функциональных требований и технических спецификаций на программное обеспечение самостоятельно путем разработки и исполнения сценариев проверки с применением методов и технологий тестирования и ревьюирования кода.

В ходе проверки работоспособности осуществляется анализ нарушений требований к программному обеспечению, принимаются решения и вносятся изменения в программный код. Программист несет ответственность за решение поставленных задач или результат деятельности группы работников. Полученные результаты представляются руководителю разработки программного обеспечения.

Рефакторинг и оптимизация программного кода осуществляется на основании функциональных требований и технических спецификаций на программное обеспечение, в том числе с использованием специализированных программных средств. Программист несет ответственность за решение поставленных задач или результат деятельности группы работников. Полученные результаты представляются руководителю разработки программного обеспечения.

Исправление дефектов, зафиксированных в базе данных дефектов сводится к воспроизведению дефектов, зафиксированных в базе данных дефектов, установлению причин возникновения дефектов, внесению изменений в программный код для устранения выявленных дефектов. Программист несет ответственность за решение поставленных задач или результат деятельности группы работников. Полученные результаты представляются руководителю разработки программного обеспечения.

Выполняются самостоятельная разработка процедур сборки модулей и компонент программного обеспечения и верификация выпусков программного продукта. Производится разработка процедур развертывания и обновления программного обеспечения, процедур миграции и преобразования (конвертации) данных и программных интерфейсов с использованием выбранных программных средств, технологий создания открытых систем. Осуществляется решение различных типов задач проектирования программных комплексов различной сложности, выбор способов реализации взаимодействия программных компонент/модулей. Требуется взаимодействие с программистами-разработчиками модулей, архитектором программного обеспечения. Полученные результаты представляются руководителю разработки программного обеспечения.

Учебно-тематический план

№	Наименование разделов	Всего часов	В том числе		Форма контроля
			Лекции	Практ. занятия	
1	**Основы языка C++**	4	2	2	Защита лабораторной работы
2	**Стандартная библиотека шаблонов языка C++**	4	2	2	Защита лабораторной работы
3	**Классы и объекты**	8	4	4	Защита лабораторной работы
3.1	Понятие класса.				
3.2	Методы и данные.				
3.3	Конструктор и деструктор				Контрольная работа
4	**Композиция, наследование и полиморфизм**	8	4	4	Защита лабораторной работы
4.1	Композиция				Контрольная работа
4.2	Виды наследования				
4.3	Полиморфизм				
4.4	Множественное наследование				
5	**Перегрузка операций**	8	4	4	Защита лабораторной работы
6	**Обработка ошибок и исключительных ситуаций**	8	4	4	Защита лабораторной работы
6.1	Способы обработки ошибок				Контрольная работа
6.2	Исключения				
6.3	Разработка кода, безопасного к возникновению исключений				
7	**Обобщенное программирование, шаблоны**	8	4	4	Защита лабораторной работы
8	**Модульное тестирование программ**	8	4	4	Защита лабораторной работы
9	**Паттерны проектирования**	16	8	8	Защита лабораторной работы
9.1	Порождающие паттерны проектирования				
9.2	Паттерны поведения				
9.3	Структурные паттерны				
	Итого	72	36	36	

Содержание курса

Тема 1. Основы языка C++

Основные языковые конструкциями языка C++ - операторы, объявление переменных, ссылки и указатели, организация ветвления, циклов, организация процедур и функций, ввод и вывод.

Тема 2. Стандартная библиотека шаблонов C++

Знакомство со стандартной библиотекой шаблонов (STL). Основные контейнеры и алгоритмы. Итераторы. Умные указатели.

Тема 3. Классы и объекты.

Понятие класса и объекта; методы, данные и свойства; ограничение доступа к данным и методам класса; создание и разрушение экземпляров класса, конструктор и деструктор; копирование и перемещение объектов.

Тема 4. Композиция, наследование и полиморфизм

Создание классов на основе имеющихся; композиция и наследование; одиночное и множественное наследование; полиморфизм; открытое, защищенное и закрытое наследование; достоинства и недостатки наследования.

Тема 5. Перегрузка операций.

Дружественные функции и классы. Перегрузка операторов сложения, вычитания, сравнения, индекс в массиве, присваивания, потокового ввода-вывода, приведения типов. Перегрузка оператора (). Функторы. Умные указатели (smart pointers).

Тема 6. Обработка ошибок и исключительных ситуаций.

Обнаружение ошибочных ситуаций. Способы обработки ошибок. Исключения в C++. Классы исключений. Разработка кода, безопасного к возникновению исключений.

Тема 7. Обобщенное программирование, шаблоны

Парадигма обобщенного программирования. Примеры обобщенных алгоритмов. Реализация механизмов обобщенного программирования в C++. Шаблоны классов и функций. Параметры шаблонов. Шаблоны и наследование.

Тема 8. Модульное тестирование, разработка через тестирование

Виды тестирования, основная идея и преимущества модульного тестирования. Разработка через тестирование (Test Driven Development, TDD), этапы разработки в стиле TDD, примеры.

Тема 9. Паттерны проектирования

Паттерны проектирования: Абстрактная фабрика, Строитель, Фабричный метод, Прототип, Одиночка, Адаптер, Компоновщик, Декоратор, Заместитель, Команда, Посетитель, Наблюдатель. Шаблон Model-View-Controller (MVC), основы программирования событийно-управляемых систем.

Методические рекомендации.

Для достижения поставленных целей преподавания курса реализуются следующие средства, способы и организационные мероприятия:

1. изучение теоретического материала дисциплины на лекциях с использованием компьютерных технологий;
2. самостоятельное изучение теоретического материала дисциплины с использованием Internet-ресурсов, информационных баз, методических разработок, специальной учебной и научной литературы;
3. закрепление теоретического материала при проведении лабораторных работ с использованием учебного и научного оборудования и приборов, выполнения проблемно-ориентированных, поисковых, творческих заданий.

При организации учебных занятий используются активные методы обучения (работа в группах, взаимообучение, самоконтроль, индивидуальные задания дифференцированной сложности).

В процессе обучения возможно использование следующих тактических технологий: лекция классическая, лекция проблемная, лекция-визуализация, лекция-диалог, аудиторно-практическое занятие классическое, практикум-лабораторная работа, самообучение.

Пособия по изучению курса

1. Б. Страуструп, Язык программирования C++, 3-е изд./Пер. с англ. --- СПб.; М.: Невский Диалект -- Издательство БИНОМ, 1999 г.

2. Б. Страуструп, Дизайн и эволюция языка C++ / Пер. с англ. -- М. ДМК, 2000.

3. С. Мейерс. Эффективное использование C++: 50 советов по улучшению ваших программ и проектов. / Пер. с англ. -- М. ДМК, 2000.

4. С. Мейерс. Наиболее эффективное использование C++: 35 новых рекомендаций по улучшению ваших программ и проектов. / Пер. с англ. -- М. ДМК, 2000.

5. Б. Эккель. Философия C++. Введение в стандартный C++/ Пер. с англ. – СПб.: Питер, 2004.

6. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C ++, 2-е изд./Пер. с англ .- М.: Издательство Бином, СПб.: Невский диалект, 1999 г.

Контрольные задания

Оценка текущей и промежуточной аттестации по курсу осуществляется по результатам выполнения лабораторных и контрольных работ. При изучении учебной дисциплины проводятся 3 контрольные работы по следующим разделам курса:

1. Классы и объекты

2. Композиция, наследование и полиморфизм
3. Обработка ошибок и исключительных ситуаций

Каждая контрольная работа контролирует освоение слушателями определенного раздела изучаемого курса. Итоговый контроль по дисциплине осуществляется по результатам выполнения лабораторных, контрольных работ и сдачи экзамена.

Экзаменационные вопросы

1. Типы данных языка C++. Целочисленные типы данных. Типы данных с плавающей запятой. Перечислимый тип.Объявление переменных и констант в языке Си++.
2. Типы данных языка C++. Структуры. Объединения. Массивы. Указатели. Ключевое слово typedef. Способы выделения памяти для хранения данных в языке Си. Статическое, автоматическое и динамическое размещение данных. Особенности и различия.
3. Операторы языка C++. Арифметические операторы, операторы отношения, логические операторы, операторы манипулирования битами.
4. Операторы инкремента и декремента. Операторы и выражения присваивания. Условное выражение. Приоритет и очередность выполнения операторов
5. Условное выражение. Инструкции и блоки. Оператор if. Оператор switch. Средства организации циклов языка C++ (for, while, do-while). Вложенные циклы. Инструкции break и continue. Инструкция goto.
6. Средства организации подпрограмм и функций в языке C++. Локальные и статические переменные функций. Аргументы функций. Передача параметров по значению, ссылке, указателю. Возвращение результата функции. Тип void. Статические функции.
7. Организация памяти в языке Си. Указатели. Взятие адреса переменной. Разыменование указателя. Инициализация указателей. Копирование указателей. Указатели и аргументы функций. Указатели на функции.
8. Массивы в языке Си++. Связь между указателями и массивами. Адресная арифметика. Строковые константы. Различия между массивами и указателями. Массивы указателей.
9. Двумерные и многомерные массивы. Указатели на указатели.Указатели на структуры. Инкремент и декремент указателя. Работа с динамической памятью в языке Си++ – функции malloc, calloc, realloc, free. Операторы newи delete.
10. Перегрузка функций. Выбор нужной функции компилятором. Разрешение неоднозначностей при перегрузке функций. Стандартные значения параметров функций. Ссылки. Инициализация ссылок. Ссылки на временные объекты. Пространства имен.
11. STL - стандартная библиотека шаблонов языка C++. Контейнеры STL (вектор, двусвязный список, множество, карта (отображение), двусторонняя очередь) Итераторы. Алгоритмы STL.
12. Основные принципы ООП. Абстракция. Инкапсуляция. Наследование. Полиморфизм. Классы и объекты. Методы, данные и свойства. Ограничение доступа к полям классам.
13. Указатель this. Константные методы класса. Изменчивые данные класса. Инициализация и деинициализация экземпляра класса. Копирование объектов. Запрещение копирования объектов.
14. Дружественные функции и классы. Дружественные операции. Статические данные и методы класса. Вложенные классы. Идиома Pimpl.

15. Композиция. Наследование. Способы наследования. Открытое, закрытое и защищенное наследование. Вызов конструкторов и деструкторов при наследовании.
16. Перегрузка методов в классе-наследнике. Виртуальные функции. Абстрактные классы. Интерфейс. Приведение типов по иерархии классов.
17. Множественное наследование. Ромбовидное наследование. Проблемы. Виртуальное наследование. Преимущества использования наследования. Недостатки использования наследования.
18. Перегрузка операций. Способы перегрузки операций. Перегрузка присваивающих выражений. Дружественные операции. Умные указатели. Перегрузка унарного плюса и минуса. Перегрузка оператора присваивания. Запрет оператора присваивания.
19. Перегрузка операций. Способы перегрузки операций. Перегрузка оператора индексации. Перегрузка операций инкремента и декремента. Перегрузка операторов потокового ввода/вывода. Перегрузка операторов приведения типа. Перегрузка оператора (). Функторы.
20. Способы обработки ошибок. Выбрасывание и перехват исключений. Разработка кода, безопасного к возникновению исключений.
21. Обобщенное программирование. Шаблоны функций. Шаблонные операторы. Шаблоны классов. Параметры шаблонов, не являющиеся типами.
22. Обобщенное программирование. Специализация шаблонов. Шаблонные методы класса. Шаблоны и наследование. Преимущества и недостатки использования шаблонов.
23. Модульное тестирование. Разработка через тестирование. Этапы разработки в стиле TDD. Преимущества использования модульных тестов и TDD. Возможные проблемы, затрудняющие модульное тестирование классов и способы их решения. Закон Деметра.
24. Паттерны проектирования. Абстрактная фабрика. Назначение. Структура. Отношения между участниками паттерна. Применение. Пример использования. Преимущества и недостатки.
25. Паттерны проектирования. Строитель. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки.
26. Паттерны проектирования. Фабричный метод. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки.
27. Паттерны проектирования. Прототип. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки.
28. Паттерны проектирования. Одиночка. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки. Особенности реализации паттерна «Одиночка» в C++.
29. Паттерны проектирования. Адаптер. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки.Адаптер класса. Адаптер объекта. Способы реализации сменных адаптеров.
30. Паттерны проектирования. Компоновщик. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки.Ссылки на родителей. Разделение компонентов. Управление дочерними компонентами.
31. Паттерны проектирования. Декоратор. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки.

32. Паттерны проектирования. Заместитель. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки. Удаленный заместитель. Виртуальный заместитель. Защищающий заместитель. Умный указатель. Оптимизация «Copy-on-write».
33. Паттерны проектирования. Команда. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки. Диаграмма взаимодействий. Поддержка отмены и повтора операций.
34. Паттерны проектирования. Посетитель. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки. Диаграмма взаимодействий.
35. Паттерны проектирования. Наблюдатель. Назначение. Структура. Применение. Пример использования. Преимущества и недостатки. Диаграмма взаимодействий. Наблюдение за несколькими субъектами. Инициирование обновления. Висячие ссылки и борьба с ними. Гарантирование непротиворечивости состояния субъекта перед отправкой уведомления. Трансляция информации о характере изменения субъекта, модель вытягивания, модель проталкивания. Явное специфицирование представляющих интерес модификаций.
36. Паттерны проектирования. Model-View-Controller. Назначение. Структура. Взаимодействие между участниками паттерна. Применение. Пример использования.

Пример задания лабораторной работы №1

Задание 1. 20 баллов

Ознакомьтесь с возможностями класса **vector** библиотеки STL.

Разработайте программу, выполняющую считывание массива чисел с плавающей запятой, разделяемых пробелами, из стандартного потока ввода в vector, обрабатывающую его согласно заданию Вашего варианта и выводящую в стандартный поток полученный массив (разделенный пробелами). В программе должны быть выделены функции, выполняющие считывание массива, его обработку и вывод результата.

В комплекте с программой должны поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	Выводимое значение
1	Прибавить к каждому элементу массива среднее арифметическое его положительных элементов
2	Каждый элемент массива должен быть умножен на минимальный элемент исходного массива
3	Умножить элементы массива, делящиеся на 3 без остатка, на среднее арифметическое элементов массива, делящихся на 2 без остатка
4	Разделить элементы массива на половину максимального элемента
5	Отсортировать элементы массива в порядке возрастания суммы составляющих их цифр
6	Умножить каждый отрицательный элемент массива на произведение максимального и минимального элементов исходного массива
7	Умножить каждый элемент массива на максимальный элемент исходного массива и разделить на минимальный элемент исходного массива
8	Прибавить к каждому элементу массива сумму трех минимальных элементов массива
9	Элементы, стоящие на четных позициях массива умножить на 2, а из элементов, стоящих на нечетных позициях, вычесть сумму всех неотрицательных элементов
10	Каждую четверку элементов массива (либо оставшуюся часть массива меньшей длины) отсортировать в порядке возрастания

Т.к. вещественные числа представляются в памяти лишь приблизительно, необходимо при подсчете суммы цифр числа принимать в расчет лишь 3 знака после запятой.

Задание 2

Ознакомьтесь с возможностями класса string (точнее, шаблона basic_string) библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	Задание	Балл
1	Разработайте функцию **std::string RemoveExtraSpaces(std::string const& arg)**, удаляющую из строки, переданной в параметре arg, лишние пробелы. Лишними считаются все пробелы в начале и конце строки, а также дополнительные пробелы между словами. Разработайте с ее использованием функцию, выполняющую удаление лишних пробелов из каждой входного потока символов и вывод результирующих строк в выходной поток.	30
2	Разработайте функцию **std::string TrimBlanks(std::string const& str)**, выполняющую отрезание пробелов в начале и в конце строки str, и возвращающую результирующую строку Разработайте на ее основе программу, выполняющую отрезание пробелов в начале и конце каждой строки, поступающей со стандартного потока ввода, и выводящую результат в стандартный поток вывода	20

Задание 3

Ознакомьтесь с возможностями контейнера std::map библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	Задание	Балл
1	Разработайте программу, выполняющую подсчет частоты встречаемости слов, поступающих со стандартного потока вывода и выводящую слова их частоты их встречаемости в стандартный поток вывода. Словом считается последовательность из одного и более символов, разделенная последовательностью из одного и более символов разделителей (пробелы, табуляции, символы конца строки). Для подсчета частоты встречаемости символов используйте отображение **слово→частота встречаемости**. **Дополнительно можно получить до 10 баллов,** если программа будет способна обнаруживать русские и английские слова, записанные в разном регистре символов, т.е. считать слова **HeLLo** и **heLLO** одинаковыми.	20

Задание 4.

Ознакомьтесь с возможностями контейнера std::set и выполните задание, соответствующее номеру Вашего варианта.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	Задание	Балл
1	Разработайте функцию **std::set<int> CrossSet(std::set<int> const& set1, std::set<int> const& set2)**, возвращающую результат пересечения[1] двух множеств целых чисел.	30

[1] Пересечением двух множеств является множество, содержащее элементы, присутствующие одновременно и в первом и во втором множестве

	С ее использованием разработайте программу, выводящую в стандартный поток вывода элементы двух множеств целых чисел и результат их пересечения. Первое множество – множество чисел от 1 до N, делящихся без остатка на сумму своих цифр. Второе множество – множество целых чисел то 1 до N, сумма цифр которых является четной. Параметр N передается пользователем в виде аргумента командной строки.	

Дополнительные задания

Задание 5.

Ознакомьтесь с возможностями класса string (точнее, шаблона basic_string) библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	Задание	Балл
1	Разработайте функцию **std::string FindAndReplace(std::string const& subject, std::string const& search, std::string const& replace)**, возвращающую результат замены всех вхождений подстроки **search** в строке **subject** на строку **replace**. В случае, если искомая строка пустая, замены строк производиться не должно. Разработайте на ее основе программу, заменяющую все вхождения искомой строки в стандартном потоке ввода на строку-заменитель и выводящую результат в стандартный поток вывода. Синтаксис командной строки: **replace.exe** <search-string> <replace-string>	40
2	Разработайте функцию **std::string HtmlEncode(std::string const& text)**, выполняющую кодирование специальных символов строки text соответствующими сущностями HTML: • “ (двойная кавычка) заменяется на ***"*** • ‘ (апостроф) заменяется на **'** • < (знак меньше) заменяется на ***<*** • > (знак больше) заменяется на ***>*** • & (амперсанд) заменяется на ***&*** Разработайте на ее основе программу, выполняющую html-кодирование текста, поступающего со стандартного потока ввода, и выводящую результат в стандартный поток вывода.	40
3	Разработайте функцию **std::string HtmlDecode(std::string const& html)**, выполняющую декодирование HTML-сущностей строки **html**, перечисленных в варианте 3, обратно в их символьное представление. Разработайте на ее основе программу, выполняющую декодирование html-сущностей текста, поступающего со стандартного потока ввода, и выводящую результат в стандартный поток вывода.	40

4	Разработайте функцию
	bool ParseURL(std::string const& url, Protocol & protocol, int & port, std::string & host, std::string & document),
	выполняющую разбор строки url, и извлечение из нее информации об используемом протоколе, номере порта, имени хоста и имени документа.
	В случае успеха функция должна возвращать true, в случае ошибки – false.
	Protocol – это перечислимый тип, задающий один из известных программе протоколов:
	enum Protocol
	{
	HTTP,
	HTTPS,
	FTP
	};
	Валидным (допустимым) URL-ом программа должна считать строку в следующем формате:
	протокол://***хост***[:***порт***][/***документ***], где
		60

	протокол – http, https или ftp (**в любом регистре**), порт – положительное число от 1 до 65535 (в квадратных скобках указаны опциональные элементы URL-а) Если порт не указан, то он должен считаться равным номеру порта по умолчанию для данного протокола (для HTTP – это 80, для HTTPS – 443, для FTP – 21). Разработайте на ее основе программу, распознающую допустимые URL-строки (разделяемые символами конца строки) в стандартном потоке ввода и выводящую в стандартный поток вывода информацию **о каждом из них** в следующем формате: **<*Исходный URL*>** **HOST: <*хост*>** **PORT: <*порт*>** **DOC: <*документ*>** Например, для URL-а http://www.mysite.com/docs/document1.html?page=30&lang=en#title должно быть выведено:	

	`http://www.mysite.com/docs/document1.html?page=30&lang=en#title`
	`HOST: www.mysite.com`
	`PORT: 80`
	`DOC: docs/document1.html?page=30&lang=en#title`	

Задание 6.

Ознакомьтесь с возможностями контейнера std::map библиотеки STL и выполните задание, соответствующее номеру Вашего варианта.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	**Задание**	**Балл**

<table>
<tr><td>1</td><td>Разработайте программу-словарь, осуществляющую перевод слов и словосочетаний, поступающих со стандартного потока ввода, с английского языка на русский с использованием заданного файла словаря и выводящую результат перевода в стандартный поток вывода.
Если вводимое слово или словосочетание, отсутствует в словаре, программа должна попросить пользователя ввести перевод и запомнить его, в случае, если пользователь ввел непустую строку.
Для выхода из диалога с программой пользователь должен ввести строку, состоящую из трех точек. Перед выходом программа спрашивает о необходимости сохранить изменения в файле словаря, в том случае, если в словарь были добавлены фразы во время текущей сессии работы с программой.
Имя файла словаря передается программе с помощью параметра командной строки. Формат файла словаря задан в приложении к лабораторной работе в каталоге tests\translator.
Пример диалога пользователя с программой:
>cat
кот, кошка
>ball
мяч
>meat
Неизвестное слово “meat”. Введите перевод или пустую строку для отказа.
>мясо
Слово “meat” сохранено в словаре как “мясо”.
>meat
мясо
>The Red Square
Неизвестное слово “The Red Square”. Введите перевод или пустую строку для отказа.
>Красная Площадь
Слово “The Red Square” сохранено в словаре как “Красная Площадь”.
>lkkvksmdv
Неизвестное слово “lkkvksmdv”. Введите перевод или пустую строку для отказа.
>
Слово “lkkvksmdv”проигнорировано.
>Тут пользователь нажимает Ctrl+Z, а затем Enter, чтобы ввести символ конца файла с клавиатуры.
В словарь были внесены изменения. Введите Y или y для сохранения перед выходом.
>y
Изменения сохранены. До свидания.
Дополнительно можно получить до 10 баллов, если программа будет способна осуществлять перевод английских слов, вводимых пользователем в произвольном регистре символов.
Например, слова CaT, при известном переводе для слова cat.</td><td>60</td></tr>
</table>

Задание 7.

Ознакомьтесь с возможностями контейнера std::set и выполните задание, соответствующее номеру Вашего варианта.

В комплекте с программой должны обязательно поставляться файлы, позволяющие проверить ее работу в автоматическом режиме.

Вариант	Задание	Балл
1	В спортивной школе обучается некоторое количество учеников. Имена учеников – уникальны. Каждый ученик посещает одну или более спортивных секций школы. У тренера каждой секции есть список учеников его секции, заданный в виде текстового файла, в котором в каждой строке записано ФИО ученика. **Задача**: на основе файлов со списками учеников каждого тренера построить список всех учеников школы. Программа принимает с командной строки имена файлов со списками учеников каждой секции (количество секций в школе, а, следовательно, параметров командной строки – произвольно) и	50

	выводит в output список всех учащихся школы. **Указания**: используйте контейнер std::set<std::string> для хранения множества имен учеников школы. В данное множество добавляйте имена учащихся, считываемых из списка учеников каждой секции. После заполнения множества выведите в output его содержимое.	
2	Для повышения культуры общения в чате необходимо написать программу-фильтр, удаляющую из сообщений участников чата недопустимые слова. Список недопустимых слов задается в текстовом файле (слова разделены последовательностью из одного и более пробелов, табуляций или символов конца строки), имя которого передается программе при помощи аргумента командной строки. Программа из каждой вводимой из стандартного потока ввода строки должна удалять присутствующие в ней недопустимые слова и выводить обработанный результат в стандартный поток ввода. Словом считается последовательность из одного и более символов, разделенных последовательностью из одного и более символов-разделителей (пробелы, табуляции, символы конца строки, знаки препинания, знаки арифметических операций, скобки). **Указания**: считайте недопустимые слова во множество строк и при обработке текста проверяйте вхождение каждого слова текста в данное множество. **Дополнительно можно получить 10 баллов**, если фильтрация слов будет производиться **с игнорированием регистра символов**, в котором они записаны. Т.е. если недопустимым словом является слово «дурак», то должны фильтроваться слова «ДуРак», «дурак», «ДУРАК» и подобные.	60