Лабораторная работа №1

«Основы объектно-ориентированного программирования»

Цель работы: изучить основы объектно-ориентированного программирования (понятия инкапсуляции, областей видимости, членов класса, наследования, полиморфизма, событийного программирования, перегрузки функции и виртуальных методов), а также ознакомиться с библиотеками для создания оконных приложений в QT C++.

Программа должна обеспечивать ввод/вывод данных с помощью визуальных компонентов, контроль ошибок при вводе, обработку исключительных ситуаций.

Задание 1.

Индивидуальные задания

Описать класс-родитель и класс-потомок, имеющие методы, указанные в соответствующем варианте задания (потомок наследует или переопределяет методы родителя и приобретает новые). Предусмотреть необходимое количество кнопок для демонстрации каждого из методов объектов.

Данные задания выполнять в соответствии с выданным вариантом:

1. Нарисовать вращающееся колесо. Родительский класс – перемещающийся круг.
2. Нарисовать повозку (прямоугольник на 2 колесах). Родительский класс – перемещающийся прямоугольник.
3. Нарисовать ракету с пламенем из сопла. Родительский класс – перемещающийся отрезок.
4. Нарисовать рожицу двигающую глазами и открывающую рот. Родительский класс – перемещающийся эллипс.
5. Нарисовать солдатика, перемещающегося и отдающего честь. Родительский класс – перемещающийся прямоугольник.
6. Нарисовать кораблик, который может поднимать флаг. Родительский класс – перемещающийся прямоугольник.
7. Нарисовать автомобиль с открывающимися дверями и включающимися фарами. Родительский класс – перемещающийся прямоугольник.
8. Нарисовать сигнальщика, подающего различные сигналы. Родительский класс – перемещающийся прямоугольник.
9. Нарисовать самосвал, который может поднимать кузов. Родительский класс – перемещающийся прямоугольник.
10. Нарисовать самолет, который может при посадке выпускает шасси. Родительский класс – перемещающийся прямоугольник.
11. Нарисовать домик, в котором открываются двери и окна. Родительский класс – перемещающийся прямоугольник.
12. Нарисовать паровоз, который выпускает дым. Родительский класс – перемещающийся прямоугольник.
13. Нарисовать воздушный шарик, который может лопнуть. Родительский класс – перемещающийся эллипс
14. Нарисовать лифт, который доставляет людей на нужный этаж. Родительский класс – перемещающийся прямоугольник.
15. Нарисовать тележку, на которой перевозят различные грузы. Родительский класс – перемещающийся прямоугольник.

Задание 2.

Реализовать иерархию классов, представляющих плоские геометрические фигуры. Необходимые фигуры: треугольник, круг, ромб, квадрат (отдельный класс), прямоугольник, звезда (пятиконечная, шестиконечная, восьмиконечная), шестиугольник и любая другая фигура не основанная на вышеперечисленных. Каждая фигура должна позволять производить следующие операции (при этом операции в общем виде для произвольной геометрической фигуры можно не реализовывать, реализовать их нужно для каждого конкретного типа фигур):

- нахождение площади и периметра, центра масс (для многоугольников можно воспользоваться триангуляцией);
- изменение координат центра масс (при этом вся фигура сдвигается на столько же, на сколько сдвинут ее центр);
- управление параметрами, специфическими для конкретных типов фигур (координаты вершин многоугольника, радиусы эллипса и т. п.);
- осуществление базовых преобразований: перемещение, поворот относительно точки, масштабирование относительно точки; при этом фигура должна изменяться плавно в течение заданного промежутка времени и на каждый такт изменения генерировать событие (можно воспользоваться Timer);
- возможность рисования фигуры на заданной канве (Canvas).

Каждый класс необходимо помещать в отдельный модуль (unit).

Создать оконное приложение, которое позволит пользователю создавать фигуры и производить с ними вышеуказанные операции.

Обязательной является реализация многоуровневой иерархии, то есть когда одни фигуры являются разновидностями других (стандартный плохой пример: прямоугольник – разновидность многоугольника, а квадрат – разновидность прямоугольника. Пример плохой т.к. принцип инвариантности не соблюдён).

Задание 3. (здесь можно получить 4)

Разработайте класс Date, хранящий день, месяц, год.

Определите и реализуйте функции-члены этого класса ():

- *Date NextDay() – определить дату следующего дня;*
- *Date PreviousDay() – определить дату предыдущего дня, bool IsLeap() – является ли год этой даты високосным, short WeekNumber() – номер недели в году для текущей даты;*
- *int DaysTillYourBithday(Date bithdaydate) – сколько дней до вашего дня рождения от текущей даты;*

- *int Duration(Date date) – количество дней между текущей и данной датой.*

Далее необходимо выполнить следующее задание: дан текстовый файл с датами (01.01.0001 24.04.2000 29.02.2000 31.12.2021). Прочитать данные в динамический массив объектов класса и для каждого элемента коллекции вычислить а) следующий день, б) разницу между текущим и следующим элементом коллекции. С помощью QT вывести на экран информацию о датах в файле таблицей. По кнопке вызывать функции класса и выводить информацию на экран. Текущий день вычислять через время из системы или использовав стандартные функции. Добавить в форму поле для ввода даты рождения пользователя. Добавить функцию изменения и добавления дат в файл без полной перезаписи и соответствующий функционал в формах. Добавить функцию открытия .txt файла из любого дискового пространства.

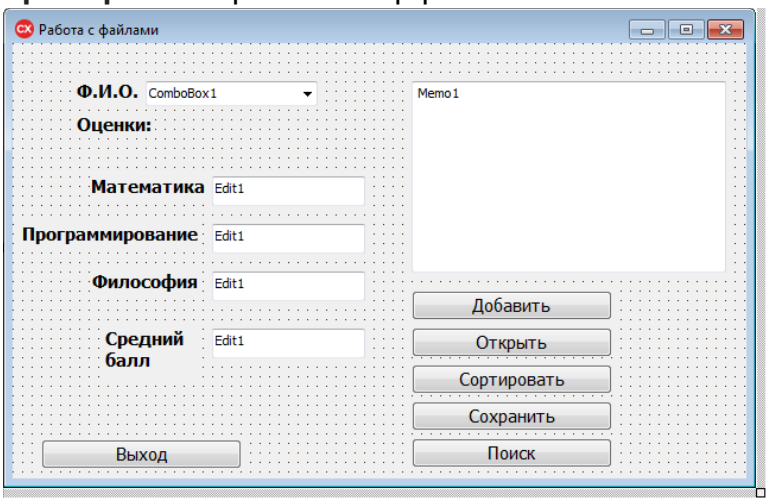
Задание 4. (здесь можно получить 6)

Разработать программу в соответствии с вариантом, реализующую работу с файлами в приложении с визуальными компонентами: Edit, Button, ComboBox, SaveDialog, OpenDialog, Memo, компонентного класса FileStream, методов LoadFromFile и SaveToFile.

Информация, обрабатываемая программой, должна храниться в файле. Использовать динамический массив объектов класса, который содержит данные в соответствии с заданием. Класс определить в отдельном модуле проекта (Add/Unit).

В каждом варианте задания реализовать следующие функции класса: инициализации данных класса, добавления, удаления, корректировки(сортировки) и просмотра записей файла.

Примерный вариант интерфейса:



Индивидуальные задачи по вариантам:

1. Магазин имеет список товаров на своем складе. Каждый пункт списка содержит следующую информацию о товаре: группа товаров, код товара, наименование продукции, наименование модели, цена, количество. Требуется:

- отсортировать товары внутри группы по одному из признаков: код товара, наименование продукции, цена;
- вывести товары, количество которых на складе меньше заданного значения;
- осуществлять поиск товара по наименованию продукции и модели.

2. Написать программу формирования ведомости об успеваемости студентов. Каждый пункт этой ведомости должен содержать: наименование специальности, номер группы, ФИО студента, оценки, полученные им за последнюю сессию, средний балл (вычисляется в программе). Требуется:

- отсортировать весь список студентов в порядке убывания среднего балла;
- отсортировать студентов в группе в порядке убывания среднего балла;
- вывести список студентов, не получивших в сессию неудовлетворительных оценок;
- осуществлять поиск студента по ФИО.

3. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждый пункт содержит следующую информацию: наименование группы изделий (телевизор, радиотелефон и т. п.), марка изделия, дата приемки в ремонт, дата исполнения заказа, состояние готовности заказа (выполнен или не выполнен). Требуется:

- выдать информацию о состоянии готовности заказов на текущие сутки по группам изделий;
- выдать информацию о заказах, невыполненных в срок;
- отсортировать заказы в группах по убыванию даты исполнения заказа;
- осуществлять поиск изделия по наименованию, дате приемки, дате исполнения заказа.

4. Разработать программу («электронную сваху») для службы знакомств. Имеется 2 списка: список женихов и список невест. В каждом списке кандидат (жених или невеста) характеризуются структурой, содержащей следующие поля: порядковый номер кандидата, данные о кандидате (имя, возраст, рост, вес, привычки, хобби), требования к партнеру (в виде диапазона Min-Max для возраста, роста, веса). Требуется:

- выявить все возможные пары с учетом требований кандидатов;
- при согласии сторон пара считается сформированной и кандидаты в списке помечаются как удаленные.

5. В фирме имеется информация о комплектующих изделиях компьютерной техники с указанием типа комплектующего изделия, фирмы-изготовителя, модели, параметров, цены, информации о наличие. Требуется:

- подобрать все возможные варианты комплектации компьютера в заданном ценовом диапазоне;
- осуществлять поиск изделия по типу изделия, фирме-изготовителю, модели;

- отсортировать изделия для каждого типа по убыванию цены.

6. В магазине имеется список поступивших в продажу CD/DVD дисков. Каждый пункт списка содержит: тип хранимой информации (фильм, музыка, СОФТ и т. п.), наименование, автора, цену и примечание. Требуется:

- сортировать внутри каждого типа информацию по наименованию либо по автору;
- осуществлять поиск диска по автору, по наименованию;
- осуществлять выбор информации по типу, по автору.

7. В деканате имеется список студентов. Каждый пункт этого списка содержит: номер группы, ФИО студента, оценки, полученные им за учебный год (две сессии: зимнюю и летнюю), средний балл по каждой сессии (вычисляется в программе). Требуется:

- сформировать список задолженников по результатам зимней сессии;
- сортировать списки по ФИО студентов внутри каждой группы;
- сформировать список студентов для отчисления (получивших две и более неудовлетворительные оценки за летнюю сессию и имевших задолженности (хотя бы одну неудовлетворительную оценку) за зимнюю сессию);
- осуществлять поиск студента в исходном списке по ФИО.

8. В больнице имеется общий список больных. Каждый пункт списка содержит: ФИО больного, пол больного, возраст, диагноз, номер палаты (при добавлении поле пустое). Требуется:

- разместить больных по палатам так, чтобы больные с одинаковым диагнозом располагались, по возможности, в одной палате. При этом **разнополых больных в одну палату размещать нельзя**. После размещения выдать список больных, которых не удалось разместить в палаты, с указанием ФИО больного, пола больного, возраста, диагноза;
- осуществлять поиск больных по номеру палаты, диагнозу, ФИО.

9. В проектной фирме есть список работ, закрепленный за сотрудниками. Каждый пункт списка содержит: название проекта, задание в рамках данного проекта, ФИО исполнителя, ФИО руководителя, дату выдачи задания, срок выполнения, дату сдачи задания сотрудником. Требуется:

- выдать список всех проектов руководителя;
- выдать список всех задач сотрудника;
- выдать список всех сотрудников проекта;
- выдать список всех сотрудников, не справившихся с заданием в срок.

10. В проектной фирме есть список работ, выполняемых сотрудником. Каждый пункт списка содержит: ФИО исполнителя, название проекта, задание в рамках данного проекта, дату выполнения задания, даты начала и окончания работы. Требуется:

- выдать список всех проектов фирмы, отсортированных по дате завершения работы;
- выдать список всех задач, работа над которыми велась одним сотрудником за текущие сутки;

- выдать список сотрудников с указанием суммарного времени работы каждого за прошедший месяц.

11. В проектной фирме есть список работ, закрепленный за сотрудниками. Каждый пункт списка содержит: название проекта, задание в рамках данного проекта, ФИО исполнителя, ФИО руководителя, дату выдачи задания, срок выполнения. Требуется:
- выдать список всех задач по конкретному проекту и их исполнителей;
- выдать список всех задач, срок завершения выполнения которых – ближайший месяц.

12. Дана рейтинговая таблица футболистов, содержащая ФИО футболиста, клуб, амплуа, количество забитых мячей, сумму штрафных очков. Требуется:
- выдать список из 10 самых успешных игроков текущего сезона, имеющих максимальное количество забитых мячей и минимальное количество штрафных очков;
- выдать список из 10 игроков с максимальной суммой штрафных очков;
- отсортировать список по убыванию количества забитых мячей.

13. Разработать программу «Биржа труда». Имеется список фирм с вакансиями. Каждый пункт списка содержит: название фирмы, должность, оклад, количество дней отпуска, требования к нанимаемому (наличие высшего образования (да/нет), возрастной диапазон (min/max); работа в данной должности не менее N-лет). Также имеется список кандидатов, каждый пункт которого содержит: ФИО кандидата, дату рождения, наличие высшего образования (да/нет), желаемую должность, минимальный оклад, список должностей, занимаемых ранее, с периодом работы. Требуется:
- для каждой фирмы подобрать возможных кандидатов по каждой вакансии;
- выдать список дефицитных должностей (кандидаты отсутствуют);
- выдать список кандидатов, для которых не найдена вакансия.

14. Разработать программу «Биржа труда». Имеется список фирм с вакансиями. Каждый пункт списка содержит: название фирмы, наименование специальности, должность, оклад, количество дней отпуска, требования к нанимаемому (наличие высшего образования (да/нет), возрастной диапазон (min/max)). Также имеется список кандидатов, каждый пункт которого содержит: ФИО кандидата, дату рождения, специальность, наличие высшего образования (да/нет), желаемую должность, минимальный оклад. Требуется:
- для каждого кандидата подобрать список возможных вакансий;
- выдать список дефицитных вакансий (кандидаты отсутствуют);
- выдать список фирм, для которых не подобран ни один кандидат.

15. В ресторане имеется меню с указанием названия блюда, его описания и цены. Также имеется список заказов за текущие сутки. Каждый пункт списка содержит: номер заказа, номер столика, название блюда, количество порций. Требуется:
- оформить счет по каждому заказу;
- выдать список блюд ресторана, пользующихся максимальным спросом;

- отсортировать список блюд по возрастанию цены.

16. В ресторане имеется меню с указанием названия блюда, его категории (холодные закуски, супы и т. п.) и цены. Также имеется список заказов за текущие сутки. Каждый пункт списка содержит: номер заказа, номер столика, название блюда, количество порций. Требуется:
- определить самое «популярное» блюдо в категории;
- определить самый «прибыльный» заказ;
- выдать список заказов по убыванию суммы заказа.

17. На фирме имеется список заказов на покупку товаров на следующие сутки. Каждый пункт списка содержит: номер заказа, адрес доставки, дата и время (от – до) доставки, вес груза в килограммах. Также имеется список курьеров, каждый пункт которого содержит: номер курьера, ФИО курьера, время работы (от – до), грузоподъемность автомобиля. Требуется:
- распределить заказы между курьерами;
- выдать список всех заказов курьера в последовательности их выполнения;
- выдать список всех заказов, которые не могут быть исполнены в срок.

18. На фирме формируются списки заказов на покупку товаров. Каждый пункт списка заказов содержит: номер заказа, дату заказа, реквизиты заказчика. Каждый заказ ссылается на список товаров в заказе, который в свою очередь содержит: номер заказа, код товара, количество товара. В прайс-листе хранятся код товара, цена за единицу товара, наименование товара. Требуется сформировать накладную по каждому заказу. Накладная содержит список товаров конкретного заказа с указанием количества и цены, а также суммарную стоимость заказа. Номер заказа и суммарная стоимость указывается в «шапке» накладной.

19. В поликлинике генерируется список талонов к врачу. Каждый пункт списка содержит: дату, время, номер очереди, ФИО больного (изначально поле пустое), номер кабинета, ФИО врача. Генерация талонов происходит на неделю, начиная с введенной даты, в соответствии с графиком работ врачей. График работы врача содержит: специализацию врача, ФИО врача, временной диапазон работы на каждый день с понедельника по субботу (длительность приема врачом больного – 15 минут). Требуется:
- сформировать списки талонов к врачу;
- осуществлять поиск всех записей к врачу на конкретную дату;
- осуществлять поиск записей о больном по ФИО.

20. В библиотеке имеется список книг. Каждый пункт списка содержит: фамилию автора (или авторов), название книги, год издания, издательство, количество страниц. Требуется:
- определить, имеются ли в данном списке книги, в названии которых
- встречается некоторое ключевое слово (например «программирование»);
- осуществлять поиск книги по ФИО автора либо по названию;
- отсортировать все книги библиотеки по году издания;
- выдать все книги одного издательства.

21. В магазине имеется список поступивших в продажу автомобилей. Каждый пункт списка содержит: страну-изготовитель, марку автомобиля, параметры автомобиля (тип двигателя, стоимость, расход бензина на 100 км, надежность (число лет безотказной работы), комфортность (в баллах). Покупатель, в свою очередь, имеет ряд требований по каждому из этих параметров. Эти требования могут задаваться в виде некоторого интервала (например, стоимость – 10…30 тыс. ($)) либо выбираться из перечисленных списков. Необходимо:

- осуществлять поиск автомобилей, удовлетворяющих требованиям покупателя;
- отсортировать автомобили по маркам, а внутри списка одной марки по любому выбранному покупателем параметру.

22. В предвыборной кампании производится регистрация кандидатов в депутаты. Каждый кандидат при регистрации, указывает номер округа, в котором он собирается баллотироваться, ФИО, наименование партии, которую он представляет, возраст, профессию, доход за прошедший год. Требуется:

- сформировать информационный бюллетень, в котором приводится следующая информация по кандидатам от каждой политической партии: число поданных заявлений на регистрацию, средний возраст кандидатов, наиболее часто встречающаяся профессия и средний доход кандидатов;
- осуществлять поиск полной информации по кандидатам каждой партии.

23. В технической службе аэропорта имеется справочник, имеющий следующую структуру: тип самолета, год выпуска, расход горючего на 1000 км. Для определения потребности в горючем техническая служба запрашивает расписание полетов на следующие сутки. Каждый пункт расписания содержит: номер рейса, пункт назначения, дальность полета, тип самолета, время вылета, время приземления в пункте назначения. Требуется:

- рассчитать суммарное количество горючего, необходимое для обеспечения полетов на следующие сутки;
- осуществлять поиск всех рейсов в конкретный пункт назначения;
- составить документ-расписание полетов.

24. В библиотеке имеется список книг. Каждый пункт списка содержит: код книги, фамилию автора, название книги, год издания, язык издания. Также имеется список читателей, каждая запись которого содержит: код читателя, ФИО читателя, домашний адрес, контактный телефон. Книги, взятые каждым читателем, заносятся в отдельный список, содержащий код читателя, код книги, дату выдачи, требуемый срок возврата и реальную дату возврата книги. Требуется:

- создать список книг, находящихся у читателей;
- создать список должников (книгу не вернули в течение десяти дней после срока возврата);
- осуществлять поиск книг по ФИО автора либо по названию;
- осуществлять поиск читателя по ФИО.

25. У администратора железнодорожных касс хранится информация о свободных местах в поездах по всем направлениям на ближайшую неделю. Данная информация представлена в следующем виде: дата отправления,

номер рейса, конечный пункт назначения, время отправления, число купейных и плацкартных мест, число свободных купейных и плацкартных мест. Требуется:

- выдать информацию об имеющихся свободных местах по каждому рейсу и каждому типу мест;
- выдать документ-расписание движения поездов по каждому дню недели;
- выдать список всех рейсов в пункт назначения с указанием общего числа свободных мест и числа проданных билетов.

26. Имеется ведомость абитуриентов, сдавших вступительные экзамены в институт. В каждой строке этой ведомости записана ФИО абитуриента, специальность, на которую он поступает, средний балл аттестата, полученные оценки по отдельным дисциплинам (например, физика, математика, русский язык). Требуется:

- выдать информацию по каждой специальности, содержащую ФИО абитуриента, суммарный балл каждого из них;
- отсортировать записи по каждой специальности по убыванию суммарного балла;
- осуществлять поиск абитуриента по ФИО.

27. В больнице имеется общий список больных. Каждый пункт списка содержит: ФИО больного, пол больного, возраст, диагноз, номер палаты, дату поступления, дату выписки. Требуется:

- указать номера палат, в которых лежат больные с более чем тремя разными диагнозами;
- сортировать списки не выписавшихся больных по убыванию даты поступления;
- осуществлять поиск больных по номеру палаты, полу, диагнозу, ФИО, возрасту (в режиме диапазона min – max).

28. Разработать программу («электронную сваху») для службы знакомств. Имеется один список женихов и невест. В списке кандидат (жених или невеста) характеризуются записью, содержащей следующие поля: порядковый номер кандидата, данные о кандидате (пол, ФИО, возраст, рост, вес), требования к партнеру (в виде диапазона min – max для возраста, роста, веса). Требуется:

- выявить все возможные пары;
- осуществлять поиск всех возможных кандидатов;
- вывести отсортированный по убыванию возраста список для кандидатов-женихов и кандидатов-невест.

29. Разработать программу формирования ведомости об успеваемости студентов. Каждый пункт этой ведомости должна содержать: наименование специальности, номер группы, ФИО студента, форму обучения (бюджетная, платная), оценки, полученные им в сессию, средний балл, вычисляемый программно. Требуется:

- отсортировать в группах студентов по ФИО;
- вывести список студентов, которые учатся на «отлично» (имеют 9- и 10-балльные оценки) с формой обучения «платная»;
- вывести списки студентов в разрезе формы обучения в порядке убывания среднего балла;

- осуществлять поиск студентов по ФИО, номеру группы, форме обучения.

30. На складе комплектующих имеется список деталей, каждая из которых характеризуется принадлежностью к группе товаров, наименованием товара, количеством, рядом признаков: признак 1 (числовой), признак 2 (строковый), признак 3 (логический). Требуется:

- отсортировать товары в группе по убыванию признака 1;
- выдавать информацию по запросу о количестве имеющихся деталей по каждому наименованию, о количестве деталей указанного наименования с заданными признаками, о количестве деталей различных наименований, имеющих одинаковые один, два или три признака.

Задание 5. (здесь можно получить 8)

Необходимо разработать иерархию классов Expression для представления арифметических выражений. Конкретнее, вам нужно определить три класса: Expression — базовый класс иерархии, Number — для представления чисел и BinaryOperation — класс описывающий бинарную операцию (+, -, * или /).

Класс Number должен хранить значение типа double.

Класс BinaryOperation должен хранить указатель на левый и правый операнды, которые сами являются арифметическими выражениями, а также тип операции (+, -, * или /), которую нужно над ними произвести.

Во всех классах должен быть метод evaluate, который возвращает значение типа double — значение соответствующего арифметического выражения, например, значение экземпляра типа Number — это число, которое он хранит, а если у вас есть объект BinaryOperation с операцией +, то нужно вычислить значения левого и правого операнда и вернуть их сумму.

В данном задании вам нужно расставить ключевое слово virtual там, где это необходимо, определить метод evaluate там, где его не хватает, а также реализовать деструкторы, там где они нужны.

При выполнении этого задания учтите, что при уничтожении объекта BinaryOperation он отвечает за уничтожение левого и правого операндов (гарантируется, что они выделены в динамической памяти).

Например, выражению 3 + 4.5 * 5 может соответствовать следующий код:

```
// сначала создаём объекты для подвыражения 4.5 * 5
Expression * sube = new BinaryOperation(new Number(4.5), '*', new Number(5));
// потом используем его в выражении для +
Expression * expr = new BinaryOperation(new Number(3), '+', sube);
// вычисляем и выводим результат: 25.5
std::cout << expr->evaluate() << std::endl;
// тут освобождаются *все* выделенные объекты
```

```
        // (например, sube будет правым операндом expr, поэтому его удалять не нужно)

        delete expr;
```

Требования к реализации: при выполнении этого задания не нужно вводить или выводить что-либо. Вы можете заводить любые вспомогательные функции, методы или классы, но не нужно реализовывать функцию main.

Пример **начальной** реализации в Приложении 1.

(Задание повышенной сложности на 9-10 баллов) *Предполагаемое решение этого задания не переносимо с точки зрения стандарта,* ***однако*** *оно проверено на различных версиях компиляторов g++/clang++/msvc.*

Вам требуется реализовать функцию, которая принимает на вход два указателя на базовый класс Expression, и возвращает true, если оба указателя указывают на самом деле на объекты одного и того же класса, и false в противном случае (т.е. если оба указателя указывают на BinaryOperation, то возвращается true, а если один из них указывает на Number, а второй на BinaryOperation, то false).

```
bool check_equals(Expression const *left, Expression const *right) {}
```

Требования к реализации: пользоваться typeid и dynamic_cast запрещено. Вызывать методы по переданным указателям запрещено.

Приложение 1

```
struct Expression
{
    double evaluate() const = 0;
};
struct Number : Expression
{
    Number(double value) : value(value) {}
private:
    double value;
};
struct BinaryOperation : Expression
{
    //Здесь op это один из 4 символов: '+', '-', '*' или '/', соответствующих операциям,
    //которые вам нужно реализовать.
    BinaryOperation(Expression const * left, char op, Expression const * right)
    : left(left), op(op), right(right) {}
private:
    Expression const * left;
    Expression const * right;
    char op;
};
```

Лабораторная работа №3
«Структуры данных Список и Стэк»

Разработать программу в соответствии с вариантом, реализующую работу со списками в приложении с визуальными компонентами.
Использовать QT C++.

Список реализовать на основе класса **(не использовать шаблоны STL и библиотеку boost)**, который содержит данные в соответствии с заданием. Класс определить в отдельном модуле проекта.

Программа должна обеспечивать ввод/вывод данных с помощью визуальных компонентов на экран и в файл который может выбрать пользователь через диалоговые окна, контроль ошибок при вводе, обработку исключительных ситуаций, алгоритм поиска элементов с конца и с начала в зависимости от текущей позиции курсора в памяти, qsort для элементов списка, вывод всех элементов списка, поиск и вывод всех элементов списка по заданному параметру (например запись 1 и запись 2 имеют одинаковое ключевое поле для поиска по элементам, соответственно оба элемента должны будут выведены на экран) с оптимизированным алгоритмом таких элементов (после сортировки списка), удаление, изменение и перезапись в файл по изменённому **полю**.

Задание 1. Реализовать два задания на двусвязном списке. Ваш вариант сделать последовательности связанных компонентов, ваш вариант +7 сделать на массиве.

1. На междугородной АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента.

Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит номер телефона и выводит все данные об этом номере;
- вводит название города и выводит данные обо всех разговорах с ним; в случае отказа от телефонного номера удаляется вся информация о нем;
- в случае добавления абонента добавляется информация о нем.

2. В ремонтной мастерской хранятся квитанции о сданных в ремонт изделиях. Каждая квитанция содержит следующую информацию: наименование группы изделий, марку изделия, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен).
Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного

списка;
- производит вывод всего списка;
- вводит номер заказа и выводит все данные об этом заказе;
- при поступлении нового заказа программа записывает его в список заказов, находящихся в исполнении;
- при исполнении заказа программа записывает его в список выполненных заказов; выводит информацию о состоянии заказов на текущие сутки по введенному наименованию группы изделий.

3. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О. абитуриента, оценки.
Составить программу, которая:
- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит фамилию абитуриента и выводит всю информацию о нем; при отсутствии абитуриента на экзамене он удаляется из списка абитуриентов;
- при наличии свободных мест обеспечивается добавление абитуриентов, сдавших экзамены в другом вузе;
- выводит список абитуриентов, средний балл которых выше среднего балла по университету.

4. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде: дата выезда, пункт назначения, время отправления, число свободных мест.

Составить программу, которая:
- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит дату, номер поезда и выводит все данные о нем;
- выводит все поезда, следующие в указанном направлении не позднее введенного времени;
- при отсутствии свободных мест на указанный поезд в нужную дату должно выводиться сообщение о невозможности выполнить заказ в полном объеме.

5. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, тип самолета, пункт назначения, время вылета.
Составить программу, которая:
- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит номер рейса и выводит все данные о нем;

- выводит всю информацию о самолетах, отправляющихся в заданный пункт назначения;
- выводит всю информацию о самолетах с указанным временем вылета; в случае отмены полетов в заданном направлении позволяет удалить из списка соответствующие рейсы, в случае открытия нового направления – добавить.

6. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О., адрес, оценки.
Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит фамилию абитуриента и выводит всю информацию о нем; выводит информацию об абитуриентах, проживающих во введенном городе; выводит информацию об абитуриентах, сдавших экзамены с результатом, не ниже введенного;
- при среднем балле абитуриента, ниже требуемого, он удаляется из списка абитуриентов;
- при наличии свободных мест абитуриент добавляется.

7. Информация о сотрудниках предприятия содержит: Ф.И.О., номер отдела, должность, дату начала работы.
Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит фамилию сотрудника и выводит всю информацию о нем;
- выводит информацию о сотрудниках, работающих во введенном отделе; выводит информацию о сотрудниках, имеющих стаж, не ниже введенного;
- обеспечивает возможность удаления из списка в случае увольнения сотрудника и добавления в список в случае трудоустройства.

8. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха.
Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит наименование продукции и выводит все данные о ней;
- выводит информацию о продукции, выпускаемой цехом с определенным номером;
- выводит информацию о цехе, выпустившем продукции в количестве не ниже введенного;

- в случае снятия продукции с производства удаляется вся информация о ней;
- в случае выпуска нового наименования продукции добавляется информация о ней.

9. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц.
Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит регистрационный номер книги и выводит все данные о ней;
- выводит информацию о книгах по введенной фамилии автора;
- выводит информацию о наличии книг по введенному названию и году издания;
- добавляет данные о книгах, вновь поступивших в библиотеку;
- удаляет данные о списываемых книгах.

10. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес.

Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит фамилию игрока и выводит все данные о нем;
- при выбытии из соревнования игрока уничтожается вся информация о нем;
- добавляется информация о новых участниках соревнований;
- выводится информация об игроках введенной весовой категории;
- выводится информация о команде, где все игроки не ниже введенного роста;
- выводится информация о самой молодой команде.

11. Информация о сотрудниках фирмы включает: Ф.И.О., табельный номер, количество проработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере.

Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;

- вводит фамилию сотрудника и выводит всю информацию о нем;
- выводит информацию о сотрудниках, имеющих зарплату не ниже введенного значения;
- выводит информацию о сотрудниках, отработавших за месяц количество часов, не ниже введенного;
- выводит информацию о сотрудниках, работавших в данном месяце сверхурочно;
- обеспечивает возможность удаления из списка в случае увольнения сотрудника и добавления в список в случае трудоустройства.

12. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия.
Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит время и выводит информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени;
- выводит информацию об автобусах, которыми можно доехать во введенный пункт назначения;
- при выезде каждого автобуса из парка вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке и записывает эти данные в список автобусов, находящихся на маршруте;
- при въезде каждого автобуса в парк вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте и записывает эти данные в список автобусов, находящихся в парке;
- по запросу выдаются сведения об автобусах, находящиеся в парке, или об автобусах, находящихся на маршруте.

13. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи.

Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит фамилию студента и выводит всю информацию о нем;
- выводит информацию о студентах, имеющих доход на члена семьи не ниже введенного значения;
- выводит информацию о студентах, имеющих доход на члена семьи выше двух минимальных зарплат и средний балл ниже введенного;
- обеспечивает возможность удаления из списка в случае отчисления студента и добавления в список в случае заселения в

общежитие на освободившееся место.

14. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад.

Составить программу, которая:

- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит наименование товара и выводит все данные об этом товаре;
- выводит информацию о товарах, хранящихся на складе больше введенного значения;
- выводит информацию о товарах, стоимость которых ниже введенного значения;
- при поступлении нового товара программа записывает его в список товаров, находящихся на складе;
- при отсутствии товара программа записывает его в список ожидаемых товаров.

15. В магазине формируется список лиц, заказавших товар. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., домашний адрес покупателя и дату постановки на учет.
Составить программу, которая:

- Удалить из списка все повторные записи, проверяя Ф.И.О. и домашний адрес;
- Обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- вводит порядковый номер заказа и выводит все данные об этом заказе; выводит информацию о заказе по введенной дате постановки на учет; выводит информацию о заказе по фамилии покупателя;
- при поступлении нового заказа программа записывает его в список ожидающих заказов;
- при исполнении заказа программа записывает его в список выполненных заказов.

Задание 2. Скобки в коде.

Вам необходимо разработать программу *используя визуальные компоненты для* проверки, правильно ли расставлены скобки в данном коде.

Вы разрабатываете текстовый редактор для программистов и хотите

реализовать проверку корректности расстановки скобок. В коде могут встречаться скобки []{}(). Из них скобки [,{ и (считаются открывающими, а соответствующими им закрывающими скобками являются],} и). В случае, если скобки расставлены неправильно, редактор должен также сообщить пользователю первое место, где обнаружена ошибка. В первую очередь необходимо найти закрывающую скобку, для которой либо нет соответствующей открывающей (например, скобка] в строке “](()”), либо же она закрывает не соответствующую ей открывающую скобку (пример: “()[}”). Если таких ошибок нет, необходимо найти первую открывающую скобку, для которой нет соответствующей закрывающей (пример: скобка (в строке “{}([]”). Помимо скобок, исходный код может содержать символы латинского алфавита, цифры и знаки препинания.

Формат входа
Поле для ввода текста или текстовый файл (рядом с полем должна быть кнопка которая предложит пользователю выбрать файл в котором будет происходит проверка), в котором находится строка состоящая из заглавных и прописных букв латинского алфавита, цифр, знаков препинания и скобок из множества []{}().

Формат выхода
Если скобки в файле расставлены правильно, выведите всплывающее окно с статусом “Success". В противном случае выведите индекс (используя индексацию *с единицы*) первой закрывающей скобки, для которой нет соответствующей открывающей. Если такой нет, выведите индекс первой открывающей скобки, для которой нет соответствующей закрывающей.

Задание 3. Использование стека для программирования алгоритма вычисления алгебраических выражений

Одной из задач при разработке трансляторов является задача расшифровки арифметических выражений, например:
r:=(a + b) * (c + d) – e;

В выражении ***a + b*** ***a*** и ***b*** – операнды, **+** операция. Такая запись называется **инфиксной** формой. Возможны также обозначения ***+ab*** – **префиксная**, ***ab+*** – **постфиксная** форма. В наиболее распространенной инфиксной форме для указания последовательности выполнения операций необходимо расставлять скобки. Польский математик Я. Лукашевич обратил внимание на тот факт, что при записи выражений в постфиксной форме скобки не нужны, а последовательность операндов и операций удобна для расшифровки, основанной на применении эффективных методов. Поэтому постфиксная запись выражений получила название **обратной польской записи (ОПЗ)**.

Например, в ОПЗ вышеприведенное выражение выглядит следующим образом:
r = ab + cd + * e –;

Алгоритм вычисления такого выражения основан на использовании стека. При просмотре выражения слева направо каждый операнд заносится в стек. В

результате для каждой встреченной операции, относящиеся к ней операнды будут двумя верхними элементами стека. Берем из стека эти операнды, выполняем очередную операцию над ними, и результат помещаем в стек. Решение задач преобразования инфиксного выражения в постфиксную запись и вычисление выражения в ОПЗ оформить в виде класса с соответствующими методами.
Вам необходимо разработать программу используя визуальные компоненты в соответствии с вариантом, реализующую расшифровку и вычисление арифметических выражений с использованием стека. *Использовать QT C++*
В основе программы должны лежать два алгоритма:
а) преобразование арифметического выражения из инфиксной формы в постфиксную (форму обратной польской записи);
б) расшифровка и вычисление выражения в постфиксной форме. Для контроля вычислить выражение обычным образом.

В окне приложения в компоненте MEMO в каждой новой строке должны быть введены прикреплённые ниже выражения в рандомном порядке и/или новое выражение введённое пользователем. В соседнем компоненте MEMO необходимо вывести результаты в любой форме в таком же порядке, как были введены выражения. Количество введёных выражений не превышает 255. Длина одного выражения не превышает 255 символов.

Выражение	*a*	*b*	*c*	*d*	*e*	Результат
1. *a*/(*b*–*c*)*(*d*+*e*)	8.6	2.4	5.1	0.3	7.9	– 26.12
2. (*a*+*b*)*(*c*–*d*)/*e*	7.4	3.6	2.8	9.5	0.9	– 81.89
3. *a*–(*b*+*c*/*d*)/*e*	3.1	5.4	0.2	9.6	7.8	2.16
4. *a*/*b*–((*c*+*d*)*e)	1.2	0.7	9.3	6.5	8.4	– 131.006
5. *a*/(*b*–*c*+*d*)/*e*	9.7	8.2	3.6	4.1	0.5	168.78
6. (*a*+*b*)*(*c*–*d*)/*e*	0.8	4.1	7.9	6.2	3.5	2.38
7. *a*/(*b*–*c*)/(*d*+*e*)	1.6	4.9	5.7	0.8	2.3	– 0.413
8. *a*/(*b*/(*c*+*d*))–*e*	8.5	0.3	2.4	7.9	1.6	1.151
9. (*a*+(*b*/*c*–*d*))*e	5.6	7.4	8.9	3.1	0.2	0.666
10. *a*/(*b*+*c*)/(*d*–*e*)	0.4	2.3	6.7	5.8	9.1	– 1.091
11. *a*–(*b*/*c*/(*d*+*e*))	5.6	3.2	0.9	1.7	4.8	– 17.51
12. (*a*–*b*)/(*c*+*d*)*e	0.3	6.7	8.4	9.5	1.2	– 0.429
13. *a*/(*b*+*c*– *d*/*e*)	7.6	4.8	3.5	9.1	0.2	1.173
14. *a*/(*b*– *c*)/(*d*+*e*)	0.5	6.1	8.9	2.4	7.3	– 0.144
15. (*a*+*b*/*c*)/(*d*– *e*)	9.1	0.6	2.4	3.7	8.5	– 2.196

Лабораторная работа №4
«Шаблоны. Итераторы. Библиотека String и Vector»

Задание 1.

Реализовать struct pair. Предусмотреть вариант pair<pair<T, T>, pair<T, T>> a;
Работа задания должна быть продемонстрирована в задании 3.

Задание 2.

Реализовать динамическую библиотеку String. Решить нижеприведённую задачу из первой лабораторной с помощью визуальных компонентов.
Реализовать и продемонстрировать с помощью визуальных компонентов следующие функции:

- `void* memcpy(void* s1, const void* s2, size_t n);`
- `void* memmove(void* s1, const void* s2, size_t n);`
- `char* strcpy(char* s1, const char* s2);`
- `char* strncpy(char* s1, const char* s2, size_t n);`
- `char* strcat(char* s1, const char* s2);`
- `char* strncat(char* s1, const char* s2, size_t n);`
- `int memcmp(const void* s1, const void* s2, size_t n);`
- `int strcmp(const char* s1, const char* s2);`
- `int strcoll(const char* s1, const char* s2);`
- `int strncmp(const char* s1, const char* s2, size_t n);`
- `size_t strxfrm(char* s1, const char* s2, size_t n);`
- `har* strtok(char* s1, const char* s2);`
- `void* memset(void* s, int c, size_t n);`
- `char* strerror(int errnum);`
- `size_t strlen(const char* s);`

*не забывайте про реализацию нужных конструкторов и операторов копирования (Правило трех).

Задача для задания 2:

Написать парсер C++ кода на языке C++.
Парсер должен выдать следующую информацию:

1. Количество переменных каждого типа и их названия. Если у переменных есть базовое значение - вывести. Базовые параметры функции не учитывать.
2. Указать сколько классов, структур, массивов было инициализировано в коде.
3. Выдать на экран список прототипов функций (для функции не имеющей прототипа также должен быть выведен прототип)
4. Выдать координату (номер строки, индекс строки) каждого изменения любой переменной, в т.ч. массива.
5. Подсчитать количество локальных переменных и выдать их координаты.
6. Подсчитать количество перегруженных функций и выдать их координаты.
7. Рассчитать глубину каждого ветвления (отсчёт с 1).
8. Вывести на экран логические ошибки которые не зависят от действий во время выполнения программы. Пример: const bool a = true; while (a){} или while (false){}.

Ввод данных сделать двумя вариантами:

1. Открыть файл .CPP
2. В MEMO вставить код.

Форматирование кода должно быть по стандарту google и предусматривать 2 варианта объявления указателей: int* p и int *p;
https://google.github.io/styleguide/cppguide.html;

Задание 3.

Решить нижеприведённую задачу из 1 лабораторной работы с помощью визуальных компонентов на самописном Vector не используя стандартные библиотеки. Реализовать динамическую библиотеку Vector (на шаблонах) и итератор для Vector. В библиотеке vector необходимо реализовать и продемонстрировать работу следующих функций с помощью визуальных компонентов:

- assign; Удаляет вектор и копирует указанные элементы в пустой вектор.
- at; Возвращает ссылку на элемент в заданном положении в векторе.
- back; Возвращает ссылку на последний элемент вектора.
- begin; Возвращает итератор произвольного доступа, указывающий на первый элемент в векторе.
- capacity; Возвращает число элементов, которое вектор может содержать без выделения дополнительного пространства.
- cbegin; Возвращает постоянный итератор произвольного доступа, указывающий на первый элемент в векторе.
- cend; Возвращает константный итератор произвольного доступа, указывающий на позицию, следующую за концом вектора.
- crbegin; Возвращает константный итератор, который указывает на первый элемент в обратном векторе.
- crend; Возвращает константный итератор, который указывает на последний элемент в обратном векторе.
- clear; Очищает элементы вектора.
- data; Возвращает указатель на первый элемент в векторе.
- emplace; Вставляет элемент, созданный на месте, в указанное положение в векторе.
- emplace_back; Добавляет элемент, созданный на месте, в конец вектора.
- empty; Проверяет, пуст ли контейнер вектора.
- end; Возвращает итератор произвольного доступа, который указывает на конец вектора.
- erase; Удаляет элемент или диапазон элементов в векторе из заданных позиций.
- front; Возвращает ссылку на первый элемент в векторе.
- insert; Вставляет элемент или множество элементов в заданную позиции в вектор.
- max_size; Возвращает максимальную длину вектора.
- pop_back; Удаляет элемент в конце вектора.
- push_back; Добавляет элемент в конец вектора.
- rbegin; Возвращает итератор, указывающий на первый элемент в обратном векторе.
- rend; Возвращает итератор, который указывает на последний элемент в обратном векторе.

- reserve; Резервирует минимальную длину хранилища для объекта вектора.
- resize; Определяет новый размер вектора.
- size; Возвращает количество элементов в векторе.
- swap; Меняет местами элементы двух векторов.

Задача для задания 3: Брюс недавно получил работу в NEERC (Numeric Expression Engineering & Research Center), где изучают и строят много различных любопытных чисел. Его первым заданием стало исследование двудесятичных чисел.

Натуральное число называется **двудесятичным**, если его десятичное представление является суффиксом его двоичного представления; и двоичное и десятичное представление рассматривается без ведущих нулей. Например, 1010 = 10102, так что **10** двудесятичное число. Числа 101010 = 11111100102 и 4210 = 1010102 не являются двудесятичными. Сначала Брюс хочет создать список двудесятичных чисел. Помогите ему найти **n**-ое наименьшее двудесятичное число.

Входные данные

Одно целое число **n** (**1** ≤ **n** ≤ **10 000**).

Выходные данные

Вывести одно число - **n**-ое наименьшее двудесятичное число в десятичном представлении.

Входные данные #1	**Выходные данные #1**
1	1
Входные данные #2	**Выходные данные #2**
2	10
Входные данные #3	**Выходные данные #3**
10	1100

Лабораторная работа №5
«Структуры данных Очереди»

Задание 1.

Создать класс **(не использовать шаблоны STL, boost)**, реализующий методы работы с очередью. Написать программу, иллюстрирующую работу всех методов работы с очередью. Результат формирования и преобразования очереди отображать в компонентах ListBox. После этого на базе родительского класса написать свой класс, реализующий метод решения своего варианта. Написать обработчик события, реализующий вызов метода решения своего варианта.

Индивидуальные задания

1. Создать очередь из случайных целых чисел. Найти минимальный элемент и сделать его первым.
2. Создать две очереди из случайных целых чисел. В первой найти максимальный элемент и за ним вставить элементы второй очереди.
3. Создать двухсвязанный список из случайных целых чисел. Удалить из списка все элементы, находящиеся между максимальным и минимальным.
4. Упорядочить элементы двухсвязанного списка случайных целых чисел в порядке возрастания методом «пузырька», когда можно переставлять местами только два соседних элемента.
5. Представить текст программы в виде двухсвязанного списка. Задать номера начальной и конечной строк. Этот блок строк следует переместить в заданное место списка.
6. Создать двухсвязанный список из случайных целых чисел. Удалить все отрицательные элементы списка.
7. Создать двухсвязанный список из случайных целых чисел. Из элементов, расположенных между максимальным и минимальным, создать первое кольцо. Остальные элементы должны составить второе кольцо.
8. Создать двухсвязанный список из случайных целых, положительных и отрицательных чисел. Из этого списка образовать два списка, первый из которых должен содержать отрицательные числа, а второй – положительные. Элементы списков не должны перемещаться в памяти.
9. Создать двухсвязанный список из строк программы. Преобразовать его в кольцо. Организовать видимую в компоненте TMemo циклическую прокрутку текста программы.
10. Создать два двухсвязанных списка из случайных целых чисел. Вместо элементов первого списка, заключенных между максимальным и минимальным, вставить второй список.
11. Создать двухсвязанный список из случайных целых чисел. Удалить из списка элементы с повторяющимися более одного раза значениями.
12. Создать двухсвязанный список и поменять в нем элементы с максимальным и минимальным значениями, при этом элементы не должны перемещаться в памяти.
13. Создать двухсвязанный список из нарисованных вами картинок. Преобразовать его в кольцо и организовать его циклический просмотр в компоненте TImage.
14. Создать двухсвязанный список из случайных чисел. Преобразовать

его в кольцо. Предусмотреть возможность движения по кольцу в обе стороны с отображением места положения текущего элемента с помощью компоненты TGauge(Kind=gkPie) и числового значения – с помощью TLabel.

15. Создать двухсвязанный список из текста вашей программы и отобразить его в TListBox. Выделить в TListBox часть строк и обеспечить запоминание этих строк. Далее выделить любую строку и нажать кнопку, которая должна обеспечивать перемещения выделенных ранее строк перед текущей строкой. При этом в TListBox должны отображаться строки из двухсвязанного списка.

Задание 2. Deque

Deque (double-ended queue) - индексируемая двусвязная очередь, поддерживающее следующие операции, каждое из которых работает за константу:

- *push_back(x)* - добавляет *x* в конец очереди.
- *push_front(x)* - добавляет *x* в начало очереди.
- *pop_back()* - удаляет последний элемент из очереди.
- *pop_front()* - удаляет первый элемент из очереди.
- *random access* индексирование.

Простейший *deque*<*T*> (здесь и далее за *T* будем считать тип данных с которым позволяет работать контейнер deque) представляет из себя некоторый массив типа *T* размера *capacity*, из которых задействовано лишь *size* элементов (размер *deque*).

При добавлении нового элемента в *deque* мы обращаемся к зарезервированным и еще не используемым элементам массива и, при отсутствии таковых, создаем новый массив размера *capacity*∗*k*, после чего можно переместить значения старого массива в новый и, наконец, добавить новый элемент. Такой подход используется и в *vector*, что позволяет обходиться без больших расходов на память.

Но что на счет итераторов? Они при такой политики будут инвалидироваться, храня указатели на элементы старого массива. Предложенный далее алгоритм позволит *resize deque* без инвалидации итераторов. Немного о процессе добавления/удаления элементов из обычного *deque*.

В любой момент времени необходимо поддерживать два, скажем так, указателя:

- первый (левый) указывает на начало очереди
- второй (правый) указывает на следующий элемент после последнего в очереди

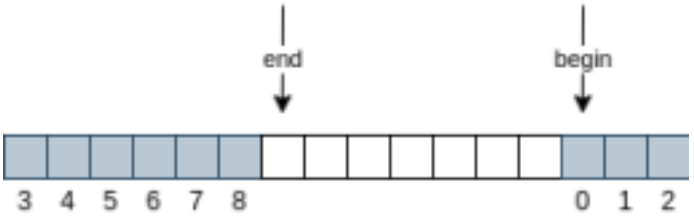
Для того чтоб добавить элемент в конец очереди достаточно положить его в ячейку массива, на которую указывает второй указатель, после чего

инкрементировать его.

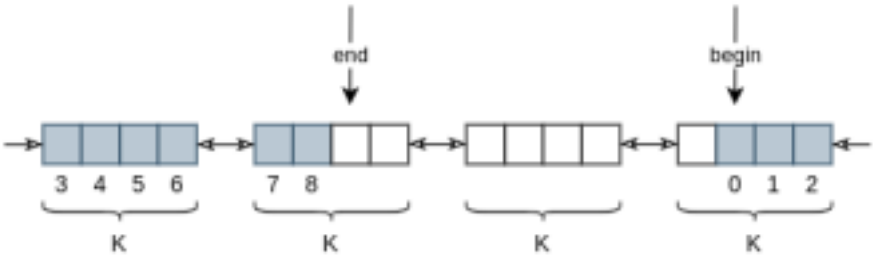
Для добавления элемента в начало очереди необходимо декрементировать левый указатель и положить туда добавляемый элемент. Делаем каждый раз *resize* при добавлении, когда *capacity==size* и по лучаем амортизированную сложность O(1).

Удаление - действия, обратные добавлению.

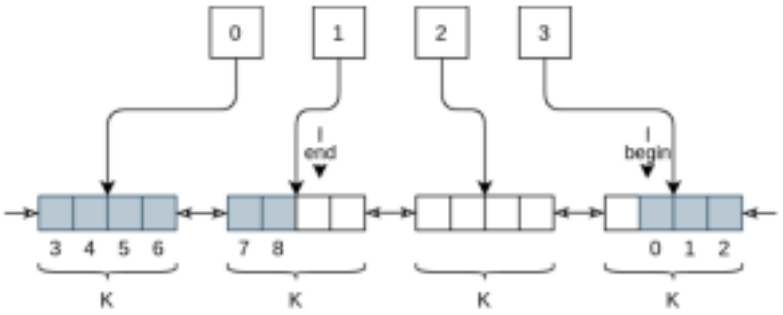
При инеркментировании/декрементировании указателей следует отме тить, что первый и последний элементы выделенного массива связаны (т.е. после инкрементирования указателя на последний элемент массива он должен указывать на первый).



Чтоб получить структуру, где итераторы не инвалидируются разобьём выделенный массив на блоки фиксированного размера *K*.



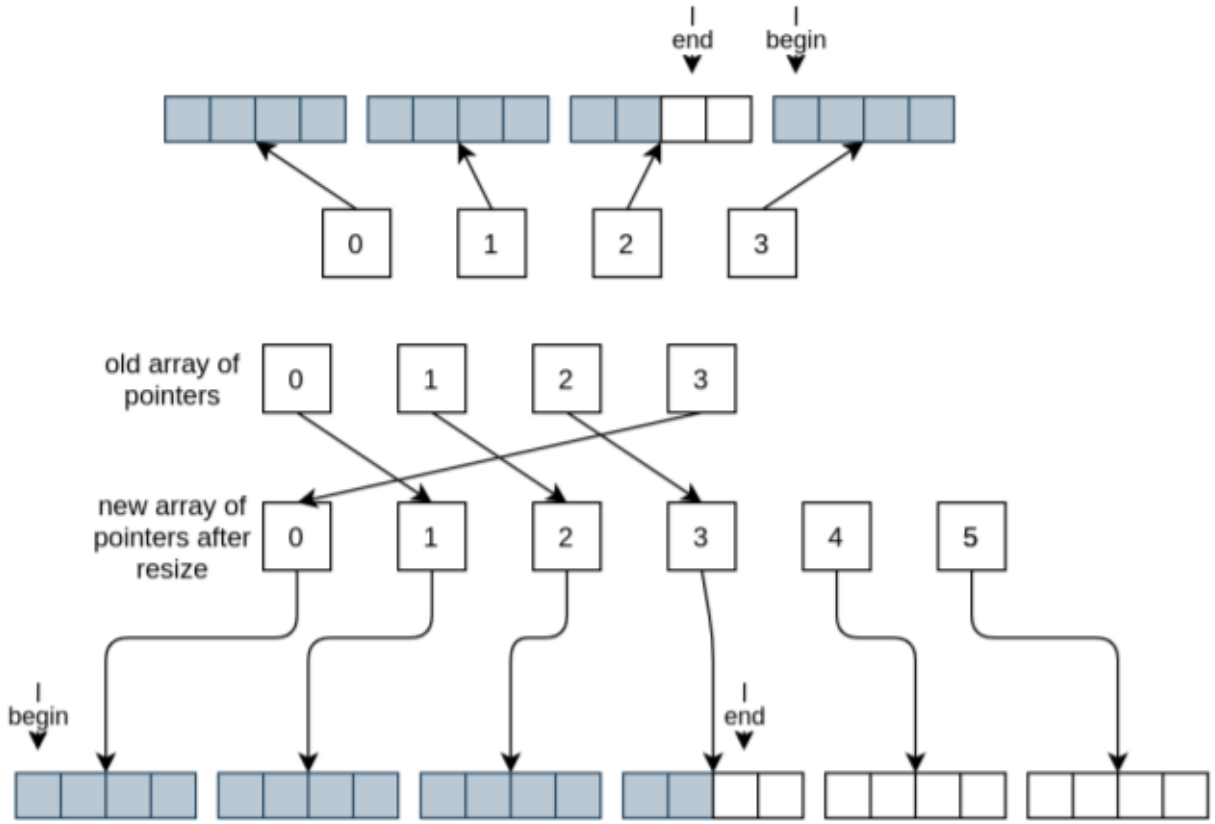
Для возможности *random access* индексирования создадим массив размера cap/k (*capacity* делится на *K*), который будет хранить указатели на блоки.



При попытке инкрементировать указатель (на объект), указывающий на последний элемент блока, то указатель в кольце из блоков перемещается на первый элемент следующего блока. Аналогично и при декременте. *resize* теперь будет проводиться над массивом указателей.

Может быть ситуация, когда при добавлении необходимо сдвинуть указатель в соседний блок со свободными элементами, однако писать туда будет нельзя, так как при *resize* тот блок, в котором находится первый

элемент очереди должен быть под первым указателем нового массива. Иными словами, начальный блок должен оказаться в начале, а конечный в конце. Но если в одном блоке получится, что *end* левее *begin*, то этот блок придется разбить, что тоже приведет к инвалидации итераторов. Потому следует наложить запрет на то, чтоб в одном блоке *end* был левее *begin*.



Задача:
Реализовать описанную структуру данных со следующими свойствами и продемонстрировать её работу при помощи визуальных компонентов:

- *push_back* - O(1)
- *push_front* - O(1)
- *pop_back* - O(1)
- *pop_front* - O(1)
- *clear*
- *size*
- *empty*
- *random access iterator* без инвалидации при *resize*, который тоже надо будет реализовать.

Лабораторная работа №6

«**Программирование с использованием деревьев. Хэширование.**»

Задание 1.

Исходная информация в виде массива находится в компоненте StringGrid. Каждый элемент массива содержит строку текста и целочисленный ключ (например, Ф.И.О. и номер паспорта).

Разработать класс для работы с деревом поиска, содержащий следующие методы **(не использовать шаблоны STL и boost)**:

- внести информацию из массива в дерево поиска;
- сбалансировать дерево поиска;
- добавить в дерево поиска новую запись;
- по заданному ключу найти информацию в дереве поиска и отобразить ее;
- удалить из дерева поиска информацию с заданным ключом;
- распечатать информацию прямым, обратным обходом и в порядке возрастания ключа.

На основе родительского класса создать производный класс для решения задачи выбранного варианта.

Написать программу, иллюстрирующую все методы работы с деревом поиска. Результат формирования и преобразования дерева отображать в компонентах TreeView и Memo. Написать обработчик события, реализующий работу с методом решения своего варианта.

Индивидуальные задания

1. Поменять местами информацию, содержащую максимальный и минимальный ключи.
2. Подсчитать число листьев в дереве. (Лист – это узел, из которого нет ссылок на другие узлы дерева.)
3. Удалить из дерева ветвь с вершиной, имеющей заданный ключ.
4. Определить максимальную глубину дерева, т.е. число узлов в самом длинном пути от корня дерева до листьев.
5. Определить число узлов на каждом уровне дерева.
6. Удалить из левой ветви дерева узел с максимальным значением ключа и все связанные с ним узлы.
7. Определить количество символов во всех строках, находящихся в узлах дерева.
8. Определить число листьев на каждом уровне дерева.
9. Определить число узлов в дереве, в которых есть указатель только на одну ветвь.
10. Определить число узлов в дереве, у которых есть две дочери.
11. Определить количество записей в дереве, начинающихся с определенной буквы (например а).
12. Найти среднее значение всех ключей дерева и найти узел, имеющий ближайший к этому значению ключ.
13. Найти запись с ключом, ближайшим к среднему значению между максимальным и минимальным значениями ключей.
14. Определить количество узлов в левой ветви дерева.
15. Определить количество узлов в правой ветви дерева.

Задание 2.

Разработать приложение, в котором содержатся следующие классы:

Родительский класс, реализующий методы работы с хеш-таблицей на основе массива стеков **(не использовать шаблоны STL и boost)**.

Производный класс, созданный на базе родительского и реализующий метод решения своего варианта.

В приложении продемонстрировать работу всех методов работы с хеш-таблицей. Результат формирования и преобразования хеш-таблицы показывать в компоненте Memo методом Print(Memo) в виде строк, отображающих стеки.

Написать обработчик события, реализующий вызов метода решения своего варианта.

Индивидуальные задания

1. Создать хеш-таблицу со случайными целыми ключами в диапазоне от–50 до +50 и преобразовать ее в две таблицы. Первая должна содержать только положительные ключи, а вторая – отрицательные.
2. Создать хеш-таблицу со случайными целыми ключами и удалить из него записи с четными ключами.
3. Создать хеш-таблицу со случайными целыми ключами в диапазоне от –10 до 10 и удалить из него записи с отрицательными ключами.
4. Создать хеш-таблицу со случайными целыми ключами и найти запись с минимальным ключом.
5. Создать хеш-таблицу со случайными целыми ключами и найти запись с максимальным ключом.
6. Подсчитать, сколько элементов хеш-таблицы со случайными ключами превышает среднее значение от всех ключей.
7. Создать хеш-таблицу из случайных целых чисел и найти в ней номер стека, содержащего минимальное значение ключа.
8. Создать хеш-таблицу из случайных целых чисел и найти в ней номер стека, содержащего максимальное значение ключа.

Лабораторная работа №7
«**Продвинутое программирование BST. Bitset.**»

Задание 1 и 2. Необходимо реализовать 2 вида BST.
Аналоги std::map и std::set. Продемонстрировать работу при помощи визуальных компонентов.
Map<KeyType, ValueType> принимает два шаблонных типа: тип ключа (KeyType), тип значения (ValueType).
В дереве данные должны лежать в парах (тот самый) pair<const KeyType, ValueType>, все операции над деревом выполнять исключительно над KeyType

Set<KeyType> ~ Map<KeyType, char> (просто фиктивное Value, которое не надо использовать).

Set, Map должны поддерживать два типа итераторов:

1. тип итераторов для итератора над вершиной дерева Node с ключом key находит следующий по, например, методу Next - должны были в midterm реализовать;
2. тип итераторов: каждая вершина дерева является еще и вершиной двусвязного списка (такого, что все ключи вершин списка упорядочены по возрастанию). Найти следующий элемент можно просто обратившись к правому соседу в списке.

Необходимо учесть, что метод Insert, который вставляет элемент в дерево, предполагает, что это дерево без итераторов **вообще**.
После этого другой виртуальный метод, для разных типов деревьев изменяет некоторые метаданные в них для работы итераторов.

Необходимо реализовать полноценный функционал хеш-таблицы (к примеру аналог std::unordered_map), а именно:
• метод contains который возвращает true если ключ X содержится в таблице
• template обязательно
• индексация аналогичная std::map (при отсутствии элемента по заданному ключу создавать его, используя конструктор по-умолчанию для ValueType). нужна версия ValueType& operator[]....., ValueType operator[](...) const
• вставка (insert), удаление (erase), clear, rehash
• хеш-таблица в качестве шаблонного аргумента обязана принимать функтор хеширования
• для самих цепочек надо использовать std::forward_list<std::pair<const KeyType, ValueType>>
• при вставке по необходимости делайте rehash
https://neerc.ifmo.ru/wiki/index.php?title=Хеш-таблица
https://neerc.ifmo.ru/wiki/index.php?title=Разрешение_коллизий

Задание 3. Реализовать класс BitSet и продемонстрировать его работу при помощи визуальных компонентов.
Кроме геттеров, сеттеров, &operator[], operator[] (index) const, необходимы следующие функции:

all	Проверяет все биты в этом параметре, bitset чтобы определить, все ли они имеют значение true .
any	Функция-член проверяет, равен ли какой-либо бит в последовательности 1.
count	Эта функция-член возвращает количество бит, заданных в последовательности бит.
flip	Инвертирует все биты в bitset или инвертирует один бит в указанной позиции.
none	Проверяет, присвоено ли хотя бы одному биту в объекте bitset значение 1.
reset	Сбрасывает все биты в bitset в значение 0 или сбрасывает бит в указанной позиции в 0.
set	Присваивает всем битам в bitset значение 1 или присваивает биту в указанной позиции значение 1.
size	Возвращает количество бит в объекте bitset.
test	Проверяет, присвоено ли биту в указанной позиции в bitset значение 1.
to_string	Преобразует объект bitset в строковое представление.

to_ullong	Возвращает сумму значений бит в bitset как unsigned long long.
to_ulong	Преобразует объект bitset в unsigned long, чтобы получить последовательность его битов, если используется для инициализации bitset.

Операции ~,&,| должны работать за O(N/16) (или 32 или 64), а другие за O(1)

Лабораторная работа №2
«Умные указатели»

Задание 1. Реализовать аналог std::shared_ptr и продемонстрировать его работу при помощи визуальных компонентов.

Задание 2. Реализовать аналог std::weak_ptr и продемонстрировать его работу при помощи визуальных компонентов.

Задание 3. Реализовать аналог std::unique_ptr и продемонстрировать его работу при помощи визуальных компонентов.