**Все лабораторные работы должны быть doxygen документированы!**

1) Шаблоны. STL Контейнеры и Алгоритмы. Функторы

Используя наработки лабораторной работы №12 (двухсвязные списки) и понятие шаблонов, сформировать класс (вместо класса - двухсвязного списка), который будет содержать коллекцию объектов, которые:

\* нечетные варианты - не содержит объекты с одинаковым строковым полем (напр. у студента - Фамилия, у стран - название...)

\* четные варианты - представляют собой частотный словарь, в котором бы указывалось, сколько раз повторяется в текущей коллекции объект с заданным строковым полем (напр. у студента - Фамилия, у стран - название...)

В новосозданном классе реализовать следующие функции:

\* вывод всех объектов на экран

\* вставка элемента в коллекцию в заданную позицию

\* удаление элемента из коллекции по ключевую строковому полю (напр. у студента - Фамилия, у стран - название...)

\* объединение двух коллекций, при этом у четных вариантов частота объектов с одинаковым строковым ключевым полем должна складываться. При этом использовать понятие "функтор"

\* сортировка элементов в коллекции по возрастанию/убыванию (в зависимости от переданного функтора) без создания промежуточной коллекции. Если это сделать нельзя - обосновать почему.

Обосновать выбор типа контейнера. При сдачи лабораторных работ знать базовые отличия между контейнерами.

http://www.yolinux.com/TUTORIALS/LinuxTutorialC++STL.html

https://cal-linux.com/tutorials/STL.html

http://www.cplusplus.com/reference/stl/

2) Классы. Наследование.

Согласно индивидуальному заданию по РГЗ, дополнить лабораторную работу №1 (STL), добавить класс-наследник, который будет расширять функционал "базового" объекта.

Расширить функционал класса-списка таким образом, чтобы он мог оперировать как объектами базового класса , так и объектами класса - наследника.

В базовом классе и классе наследнике переопределить операторы вывода. В классе-списке создать (если до этого отсутствовал) функцию вывода всех

элементов на экран (как базовых так и наследников). Продемонстрировать работоспособность данной функции.

В среде Visual Studio создать UML диаграмму классов для наглядного отображения связи созданных классов.

http://cppstudio.com/post/10103/

http://www.quizful.net/post/Inheritance-in-C++

https://code-live.ru/post/cpp-class-inheritance/

http://alenacpp.blogspot.com/2006/03/blog-post\_11.html

3) Обработка исключительных ситуаций

На базе лабораторной работы №2 (наследование), расширить функционал:

- в классе Списка переопределить оператор индексирования для доступа к i-му элементу списка. Обработать ситуацию доступа к несуществующему элементу

списка при помощи механизма исключительных ситуаций, используя свой собственный тип исключительной ситуаций

- изменить методы установки, получения элементов из списка таким образом, чтобы при передачи несуществующего индекса "выбрасывалась" ошибка

- в методе main предусмотреть возможность "отлавливания" и соответствующей обработки исключительных ситуаций

http://valera.asf.ru/cpp/book/c11.html

http://www.codenet.ru/progr/cpp/Try-Catch-Throw.php

https://msdn.microsoft.com/ru-ru/library/hh279678.aspx

https://msdn.microsoft.com/ru-ru/library/4t3saedz.aspx

4) Препроцессорная обработка.

Используя наработки предыдущих лабораторных работ, при помощи макроопределений сделать следующую "кастомизацию":

- четные варианты: в начале и конце каждой функции проверять, определенена ли переменная/макрос DEBUG. Если такова присутствует, то вывести на экран

1) текущую дату и время

2) имя текущего файла и текущей функции

3) по окончанию функции также вывести время выполнения функции

- нечетные варианты:

создать макроопределение (некий PRINT\_TYPE) в зависимости от которого новосозданная функция PRINTME (строка), будет выводить данные

а) на консоль при помощи функции cout

б) в файл, в заранее известный файл при помощи fostream

в) на консоль при помощи функции printf

г) в файл, в заранее известный файл при помощи fprintf

http://www.redov.ru/kompyutery\_i\_internet/spravochnoe\_rukovodstvo\_po\_c/p16.php

http://bibliofond.ru/view.aspx?id=7263

http://www.cplusplus.com/doc/tutorial/preprocessor/

http://www.tutorialspoint.com/cplusplus/cpp\_preprocessor.htm

http://www.codingunit.com/cplusplus-tutorial-preprocessor-directives

5) Разработка статических и динамических .dll библиотек.

На основе ранее разработанного функционала по работе с прикладной области, сформировать два типа типа библиотек (статическую и динамическую).

Продемонстрировать работу / использование разработанных библиотек.

Продемонстрировать использование собственно-определенных констант (напр. DEBUG\_MODE), определённых в лабораторной работе №4.

Заранее продумать поведение функций таким образом, чтобы их можно было бы использовать как при работе с консолью, так и в других случаях (напр. при работе с "окнами" (см. лаб №7-8)

Библиотека должна быть 100% doxygen документирована! В качестве результатов выполнения, предоставить результаты выполнения тестов, структуру библиотеки, одну из страниц документации.

Linux/gcc:

http://www.firststeps.ru/linux/r.php?3

https://younglinux.info/book/export/html/381

http://www.linuxfocus.org/Russian/November1997/article6.html

http://mech.math.msu.su/~zubr/library.html

Windows/MSVS:

https://msdn.microsoft.com/ru-ru/library/ms235627.aspx

https://msdn.microsoft.com/ru-ru/library/ms235636.aspx

6) Проверка своего кода путем написания доп. функций их проверки (модульное тестирование без использования сторонних библиотек). В самой лабораторной они должны понять принцип.

Более сложные вещи по проверки они должны будут осознать при оформлении РГЗ. Данная работа не предполагает разработки фреймворка.

Лишь написания функций сравнения ожидаемого и полученного результата и в функции main их использование (Строить свой аналог фреймворка по тестированию не надо!!! Использовать внешние фреймворки не обязательно!!! )

Обновить код проекта лабораторной работы №5, разработав функции по проверке / верификации всех публичных методов/функций в разработанной библиотеке.

В результате выполнения каждого теста , должна выводиться информация о том, какой тест выполняется (имя метода-теста) + статус верификации. В случае, если верификация не прошла, необходимо вывести информацию о том, что именно не прошло проверку.

Написанные тесты не должны быть включены в состав библиотеки!

https://msdn.microsoft.com/ru-ru/library/hh598953.aspx

http://habrahabr.ru/post/169381/

http://habrahabr.ru/post/133315/

http://hitfounder.livejournal.com/21424.html

https://netbeans.org/kb/docs/cnd/c-unit-test\_ru.html

7-8) Разработка консольных приложений с созданием простого окна и отслеживанием событий WMPaint. Данная тема должна затронуть базовую работу с оконными приложениями.

Используя наработки существующих лабораторных (в частности библиотеку из л/б #5 для разработанной прикладной области), создать графическое представление разработанного функционала.

В качестве средств создания окон, обработчиков событий, и т.д. использовать средства WinAPI/ Разработанное приложение должно уметь выполнять следующие действия:

- создание объекта заданного типа (родитель / наследник)

- удаление объекта / объектов по заданному критерию

- редактирование выбранного объекта

- вывод на экран всех объектов

- сохранение всех объектов в файл

- восстановление всех объектов из файла

- [дополнительный функционал по желанию]

http://cppstudio.com/post/9741/

http://cppstudio.com/post/9843/

http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html

http://window.edu.ru/catalog/pdf2txt/051/24051/6561?p\_page=6