МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ

по учебной вычислительной (ознакомительной) практике

Большакова Артёма Викторовича

студента 1 курса, 2 группы

специальность «Информатика»

Руководитель практики:

старший преподаватель

О. М. Кондратьева

Минск, 2024

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc170684846)

[Введение 3](#_Toc170684847)

[Глава 1 Теоретические основы 4](#_Toc170684848)

[1.1 Общая характеристика курса «Контроль версий с помощью Git» 4](#_Toc170684849)

[1.1.1 Концепт контроля версий 4](#_Toc170684850)

[1.1.2 Установка и настройка 4](#_Toc170684851)

[1.1.3 Создание и управление репозиторием 4](#_Toc170684852)

[1.1.4 Внутренне устройство git и отмена действий 4](#_Toc170684853)

[1.1.5 Ветвление 4](#_Toc170684854)

[1.1.6 Дополнительные сведения 4](#_Toc170684855)

[1.2 Общая характеристика курса «Python» 5](#_Toc170684856)

[1.2.1 Модули курса 5](#_Toc170684857)

[1.3 Выводы к главе 1 5](#_Toc170684858)

[Глава 2 Практическая часть курса 6](#_Toc170684859)

[2.1 Решение задач 6](#_Toc170684860)

[2.2 Дополнительный материал 6](#_Toc170684861)

[2.3 Выводы к главе 2 6](#_Toc170684862)

[Глава 3 Индивидуальный проект 7](#_Toc170684863)

[3.1.1 Класс MyList 7](#_Toc170684864)

[3.1.2 Интерфейс 7](#_Toc170684865)

[3.1.3 Фрагменты кода с реализацией 8](#_Toc170684866)

[3.2.1 Класс MyStack 8](#_Toc170684867)

[3.2.2 Интерфейс и фрагменты кода с реализацией 8](#_Toc170684868)

[3.2.3 Использование MyStack для графического отображения стека 9](#_Toc170684869)

[3.3.1 Итератор 9](#_Toc170684870)

[3.3.2 Интерфейс ListIterator и фрагменты кода 9](#_Toc170684871)

[3.4 Графический интрефейс 10](#_Toc170684872)

[3.5 Выводы к главе 3 11](#_Toc170684873)

[Заключение 12](#_Toc170684874)

[Список использованных источников 13](#_Toc170684875)

# Введение

Целью учебной вычислительной (ознакомительной) практики является закрепление полученных знаний за первый год обучения посредством выполнения специальных учебных заданий, участия в работе над командным проектом.

Задачи практики:

* Знакомство с IT компанией, возможностями обучения и повышения квалификации, знакомство с разными направлениями профессиональной подготовки IT специалистов.
* Изучение инструментов для совместной разработки проектов и контроля версий.
* Подготовка и настройка программных сред и средств тестирования для прохождения тренинга по выбранному направлению.
* Выполнение стандартных заданий.
* Выполнение заданий повышенного уровня.
* Оформление отчета.

# Глава 1 Теоретические основы

## 1.1 Общая характеристика курса «Контроль версий с помощью Git»

Курс «Контроль версий с помощью Git» предоставляет важные моменты и детали работы с системой контроля версий Git, которые помогают в освоении данного инструмента.

## 1.1.1 Концепт контроля версий

Перечислены основные преимущества контроля версий, а также примеры практического использования для создания высокоуровневых программ.

## 1.1.2 Установка и настройка

Подробная инструкция по скачиванию и установки системы контроля версий Git, а также подробно рассказаны основные моменты по настройке git.

## 1.1.3 Создание и управление репозиторием

Описан процесс создания репозитория. И представлены основные команды для фиксирования этапов разработки программы.

## 1.1.4 Внутренне устройство git и отмена действий

Описание того, как устроен git, в частности директория git, описано и показано использования файла .gitignore для фиксирования только нужных нам файлов. И показано использование команд для отмены, отката действий.

## 1.1.5 Ветвление

Рассказано про ветки: что это и для чего используются. Показано как использовать команды для создания/объединение веток.

## 1.1.6 Дополнительные сведения

Описание основных стратегий по устранению конфликтов при слиянии веток. Рассказано про то как и для чего использовать теги. А также описано как использовать git для синхронизации версий с удаленных машинах.

## 1.2 Общая характеристика курса «Python»

## 1.2.1 Модули курса

Перечислить в подпунктах 1.2.1, 1.2.2 и т.д модули, которые были изучены. Здесь следует перечислить и кратко описать разделы изученного материала со ссылками на источники. Приводите фрагменты разработанного кода. Пример:

Работу условных операторов языка Python можно продемонстрировать на примере задачи: «Пусть от пользователя требуется ввести два целых числа: температуру на улице вчера и сегодня. А программа ответит – сегодня теплее, холоднее или же температура не изменилась»:

yesterday\_temp = int(input())

today\_temp = int(input())

if today\_temp > yesterday\_temp:

print("Сегодня теплее, чем вчера.")

elif today\_temp < yesterday\_temp:

print("Сегодня холоднее, чем вчера.")

else:

print("Сегодня такая же температура, как вчера.")

## 1.3 Выводы к главе 1

Текст выводов.

Выводы и предложения должны быть конкретными, реальными и обоснованными, вытекать из полученных результатов. Выводы пишутся тезисно (по пунктам).

# Глава 2 Практическая часть курса

## 2.1 Решение задач

Перечислить пройденные решенные задачи, тесты, полученные оценки. Для каждой задачи в краткой форме – постановка задачи на русском языке, фрагмент кода решения, результат тестирования, дополнительное описание (при необходимости). В заключительном отчете допускается отсутствие скриншотов, подтверждающих полученную оценку тестирования, если таковые были в предварительном отчете, размещенном на edufpmi.

## 2.2 Дополнительный материал

Перечислить выполненные дополнительные работы

## 2.3 Выводы к главе 2

Текст выводов.

Выводы и предложения должны быть конкретными, реальными и обоснованными, вытекать из полученных результатов. Выводы пишутся тезисно (по пунктам).

# Глава 3 Индивидуальный проект

В данной главе представлен разработанный проект (при наличии).

Описать самостоятельно реализованный проект (проекты): назначение, фрагменты кода с комментариями, копии экранов.

## 3.1.1 Класс MyList

Класс MyList представляет собой реализацию односвязного списка. Односвязный список — это структура данных, состоящая из элементов-узлов. Каждый узел состоит из ссылки на следующий узел списка и данные. Последний узел, не имеющий следующего узла, указывает на nullptr.

## 3.1.2 Интерфейс

Интерфейс моей реализации MyList включает следующие методы:

* MyList() : конструктор по умолчанию, присваивает указателю на первый элемент списка значение nullptr.
* MyList(int a) : Конструктор с одним параметром, присваивает единственному узлу значение a и указание на nullptr.
* MyList(int a, int b, int c) : конструктор с тремя параметрами, создает 3 узла, присваивает им значения данных a, b и c, при этом первый узел со значением a указывает на узел со значением b, а уже этот узел указывает на 3 узел со значением c, который уже указывает на nullptr.
* ~MyList() : деструктор, вызывается при удалении объекта или при его выходе за пределы области видимости, поэлементно удаляет из памяти каждый узел MyList.
* bool Empty() : возвращает значение true или false, если список пуст и если список имеет хоть один элемент соответственно.
* void AddFirst(int value) : добавляет в начало списка новый элемент.
* int DeleteElement() : удаляет из списка первый элемент и возвращает хранившееся в нем значение.
* void Print() : Выводит содержимое списка в порядке от первого элемента до последнего.
* int Count() : Возвращает количество элементов в списке, если он пустой выводит 0.
* int Get(int n) : Выводит значение элемента списка находящегося на позиции n.
* void ToVector(std::vector<int>&) : принимает указатель на вектор, в который будут записаны значения элементов в списке в порядке от первого до последнего.

## 3.1.3 Фрагменты кода с реализацией

Фрагменты реализации конструкторов

MyList::MyList()

{

first = nullptr;

}

MyList::MyList(int a)

{

first = new SElement(a, nullptr);

}

MyList::MyList(int a, int b, int c)

{

first = new SElement(a, nullptr);

SElement \*temp = new SElement(b, nullptr);

first->next = temp;

temp = new SElement(c, nullptr);

first->next->next = temp;

}

Остальные реализации можно увидеть в git репозитории[https://github.com/ArpiJokle/AB\_Practice/tree/master/MyList]

Пример использований класса MyList [рис. 1]

## 3.2.1 Класс MyStack

“Адаптер – паттерн, структурирующий классы и объекты” [Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. — СПб.: Питер, 2020]

Адаптеры используются для преобразования интерфейса одного класса в другой интерфейс. Адаптеры обеспечивают совместную работу классов, у которых несовместимые интерфейсы.

Класс MyStack это класс-адаптер, который приватно наследуется от класса MyList. Другими словами, MyStack наследует функционал MyList но скрывает его от внешнего доступа. MyStack был реализован пользуясь советами из [Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. — СПб.: Питер, 2020].

## 3.2.2 Интерфейс и фрагменты кода с реализацией

Интерфейс класса MyStack:

* Конструкторы и деструктор наследованы из MyList
* int Pop() : извлекает элемент из стека
* void Push(int value) : добавляет элемент в стек
* bool IsEmpty() : проверяет пустой ли стек
* void PrintStack() : Выводит содержимое стека
* void ToVector(std::vector<int>&) : преобразует стек в вектор

Реализация

int MyStack::Pop() {return DeleteElement();}

void MyStack::Push(int a) {AddFirst(a);}

bool MyStack::IsEmpty() {return Empty();}

void MyStack::PrintStack() {Print();}

void MyStack::toVector(std::vector<int> &Vec) {ToVector(Vec);}

Полный код реализации MyStack можно увидеть на git репозитории [https://github.com/ArpiJokle/AB\_Practice/tree/master/MyStack]

## 3.2.3 Использование MyStack для графического отображения стека

Графическая визуализация класса MyList с помощью класса Painter в Qt [рис. 2]

## 3.3.1 Итератор

“Итератор – паттерн поведения объектов” [Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. — СПб.: Питер, 2020]

Итераторы позволяют реализовать последовательное обращение ко всем элементам составного объекта, не раскрывая внутреннее представление

## 3.3.2 Интерфейс ListIterator и фрагменты кода

ListIterator это класс-итератор для класса MyList, реализованый наследованием от класса Iterator и класса MyList и имеющий следующий интерфейс:

* ListIterator(MyList \*aList) : Конструктор итератора
* void First() : позиционирует итератор на первый элемент MyList
* void Next() : делает текущим следующий элемент
* bool IsDone() : проверяет выходит ли итератор за пределы MyList
* int CurrentItem() : возвращает значение текущего элемента MyList

ListIterator::ListIterator(

MyList \*aList) : \_list(aList), \_current(0) {}

Реализация методов класса ListIterator

void ListIterator::First() {\_current = 0; }

void ListIterator::Next() {\_current++;}

bool ListIterator::IsDone() {return \_current <= \_list->Count();}

int ListIterator::CurrentItem() {return \_list->Get(\_current); }

Полный код реализации ListIterator можно увидеть на git репозитории [https://github.com/ArpiJokle/AB\_Practice/tree/master/MyListIterator]

Примеры использования [рис. 3]

## 3.4 Графический интрефейс

Пример использования MyList

|  |
| --- |
|  |
| Рис.1 – Пример использования MyList |

Пример использования MyStack

|  |  |
| --- | --- |
|  | |
| Рис.2 – Пример использования MyStack |

Пример использования ListIterator

|  |
| --- |
|  |
| Рис. 3 – Пример использования ListIterator |

## 3.5 Выводы к главе 3

Текст выводов.

Выводы и предложения должны быть конкретными, реальными и обоснованными, вытекать из полученных результатов. Выводы пишутся тезисно (по пунктам).

# Заключение

В общем случае заключение должно содержать:

* краткие выводы по результатам **индивидуальной** работы;
* оценку количества и полноты решений, поставленных задач по каждому модулю (уровню сложности и т.п.).

Недопустимо искажение результатов или умолчание отрицательных результатов.

Помимо оценки результатов работы, заключение по работе может содержать информацию о пути и целях дальнейшей работы.

# Список использованных источников

1. Усов, В. Swift. Разработка приложений под iOS на основе фреймворка UIKit // В. Усов. – Москва, 2021. – 492 с. (Книга с одним автором)
2. Дробышевский, Н. П. Ревизия и аудит : учеб.-метод. пособие / Н. П. Дробышевский. – Минск : Амалфея : Мисанта, 2013. – 415 с. (Книга с одним автором)
3. Дьяченко, Л. С. Методические рекомендации по подготовке и сдаче государственного экзамена по педагогике / Л. С. Дьяченко, Н. К. Зинькова, Р. В. Загорулько. – Витебск : Витеб. гос. ун-т, 2013. – 53 с. (Книга с тремя авторами)
4. Национальный статистический комитет Республики Беларусь [Электронный ресурс]. – Режим доступа: http://belstat.gov.by/. – Дата доступа: 24.06.2016. (Сайт целиком, т.е. url заканчивается доменом первого уровня)
5. Полоник, С. С. Теоретико-методологические основы обеспечения внешнеэкономической безопасности Республики Беларусь в условиях усиления международной конкуренции / С. С. Полоник, Э. В. Хоробрых, А. А. Литвинчук // Вестн. Полоц. гос. ун-та. Сер. D, Экон. и юрид. науки. – 2013. – № 5. – С. 7–16. (Статья из журнала)
6. Жданухин, Д. Ю. Коллекторские агентства: основные черты деятельности, история появления и зарубежный опыт [Электронный ресурс] / Д. Ю. Жданухин // Юридическая Россия : Федер. правовой портал. – Режим доступа: http://law.edu.ru/doc/document.asp?docID=1233648. – Дата доступа: 22.06.2016. (Статья с сайта с указанием автора)
7. О демографической ситуации в январе – июне 2016 года [Электронный ресурс] // Национальный статистический комитет Республики Беларусь. – Режим доступа: http://www.belstat.gov.by/o-belstate\_2/novosti-i-meropriyatiya/novosti/o\_demograficheskoy\_situatsii\_v\_yanvare\_iyune\_2016\_g/. – Дата доступа: 29.07.2016. (Статья с сайта без автора)