ФГБОУ ВО

«БЕЛГОРОДСКИЙ ГОСДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ ИМ В.Я. ГОРИНА»

Лабораторная работа №1

По дисциплине: «Программная инженерия»

На тему: «Система контроля версий git»

Выполнила студент:

Рекунов Р.Д

Группы 28 ПИ

Проверил:

Ващенко Р.А.

Майский 2022

Лабораторная работа №1

«Система контроля версий git»

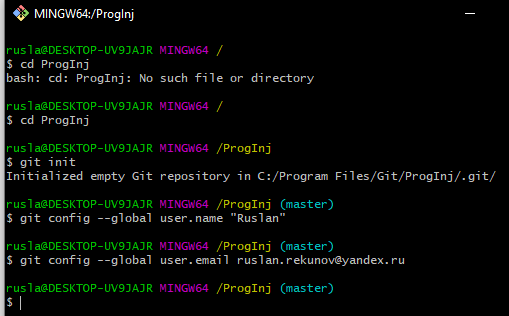
**Цель работы:**

1. Изучить на практике понятия и компоненты систем контроля версий (СКВ), приемы работы с ними.
2. Освоить специализированное ПО и распространенный сервис для работы с распределенной СКВ Git — git-bash и GitHub.com.

Ход работы:

**Задание на лабораторную работу**

1. Отработать навыки использования хранилища на локальной машине:
   1. Настроить Git, указав имя и e-mail разработчика для подписи commit-ов.

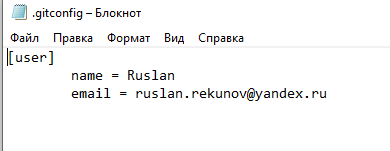


Для перехода в папку репозитория и для работы с ней, необходимо ввести команду cd «Название папки», в данном случае «cd ProgInj».

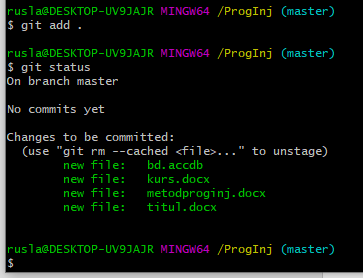
Созданную папку необходимо подключить к системе контроля версий, для этого вводится команда «git init».

Для создания своего глобального пользователя, необходимо указать имя и почту. Чтоб указать имя, необходимо ввести команду «git config –global user.name «Ruslan»». Для указания почты, «git config –global user.email [ruslan.rekunov@yandex.ru»](mailto:ruslan.rekunov@yandex.ru).

Это файл - конфиг с данными репозитория (подписи):



Добавляем в репозиторий свои файлы:

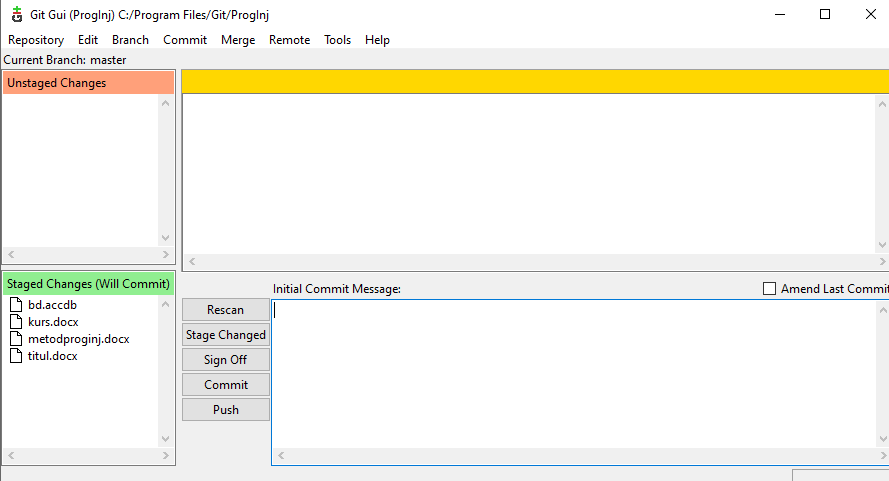


Команда «git add .» добавляет все файлы для коммита, вместо точки можно добавлять по отдельности вводя наименование файлов которые необходимы.

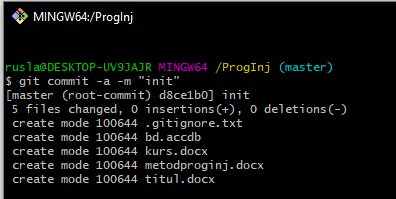
С помощью команды «git status» проверить статус наших файлов, загружены они в репозиторий или нет.

* 1. Создать хранилище для учебного проекта.

Открываем созданный репозиторий в хранилище Git Gui.



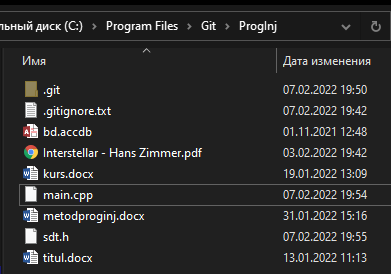
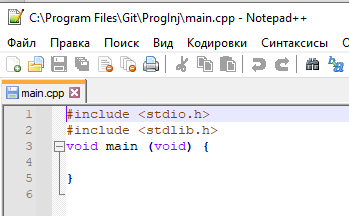
* 1. Совершить несколько commit-ов.



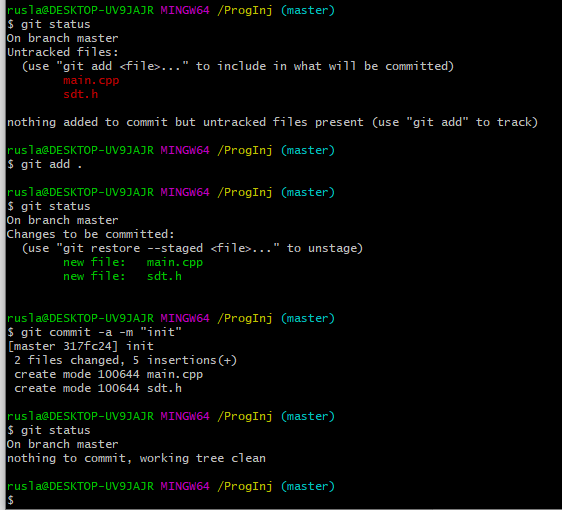
C помощью команды «git commit –a –m «init»» производим коммит и фиксируем файлы в репозиторий.

* + 1. Скопировать sdt.h в каталог хранилища и создать файл main.cpp с включением sdt.h и пустой функцией main().

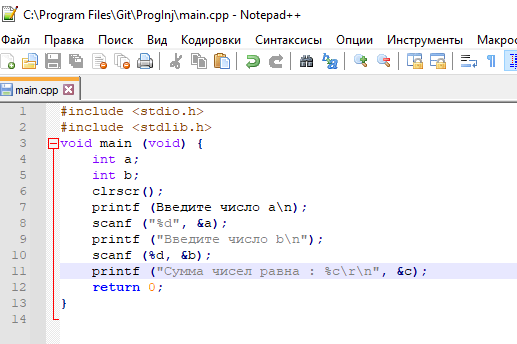
Для создания кода подключаем заголовочные файлы и прописываем тип переменной «void» (пустота) не заполняет память компьютера и «main» является главной функцией.



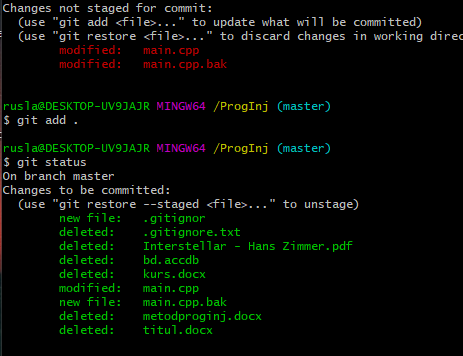
Делаем коммит и фиксируем в репозиторий новые два файла.

.

1. Добавить в программу ввод двух целых чисел с приглашением.



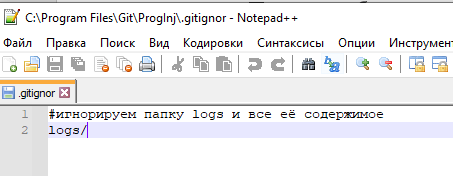
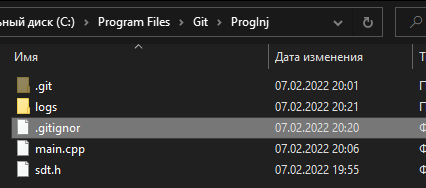
Совершаем изменения коммит.



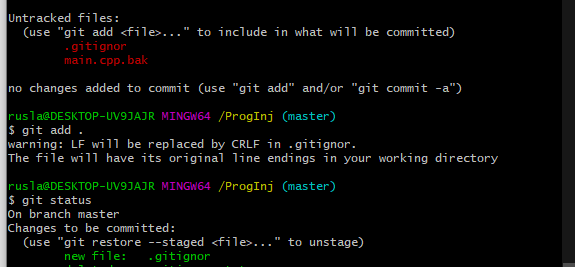
1.4. Предотвратить автоматическое добавление в хранилище файлов, не нуждающихся в контроле версий, — \*.o и \*.exe.

Правило об игнорировании следует помещать в файл .gitignore в корневом каталоге хранилища («.gitignore in repository root»). Этот файл также попадает под контроль версий, поэтому после создания правил требуется совершить commit изменений в файле. gitignore.

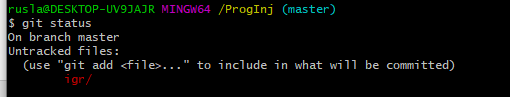
Создаем файл. gitignor



И совершаем изменения в коммите.

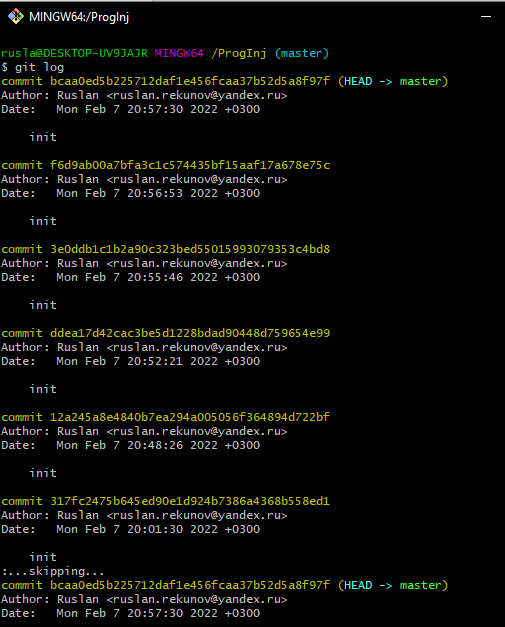


Игнорируемые файлы:

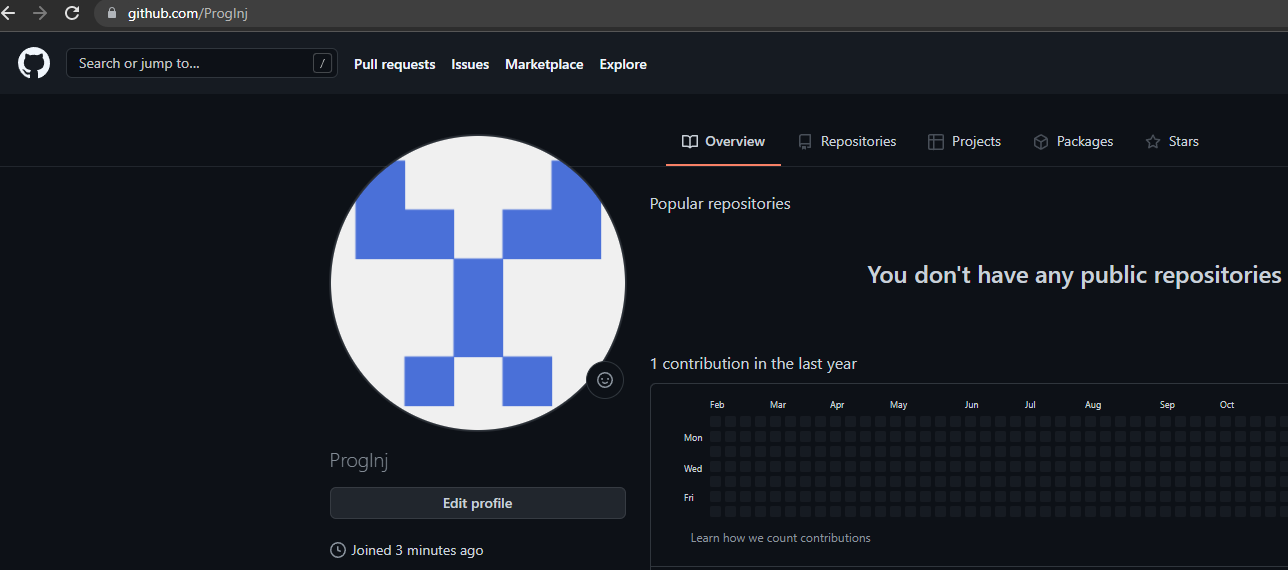


* 1. Добавить в программу вывод суммы введенных чисел и совершить commit.
  2. Просмотреть историю (журнал) хранилища.

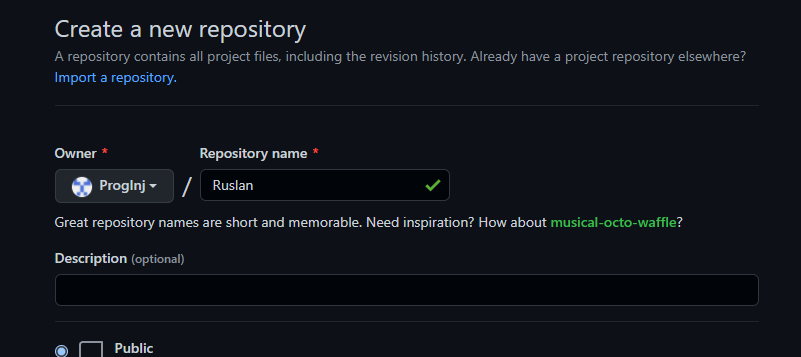
Чтобы посмотреть историю необходимо ввести команду «git log»



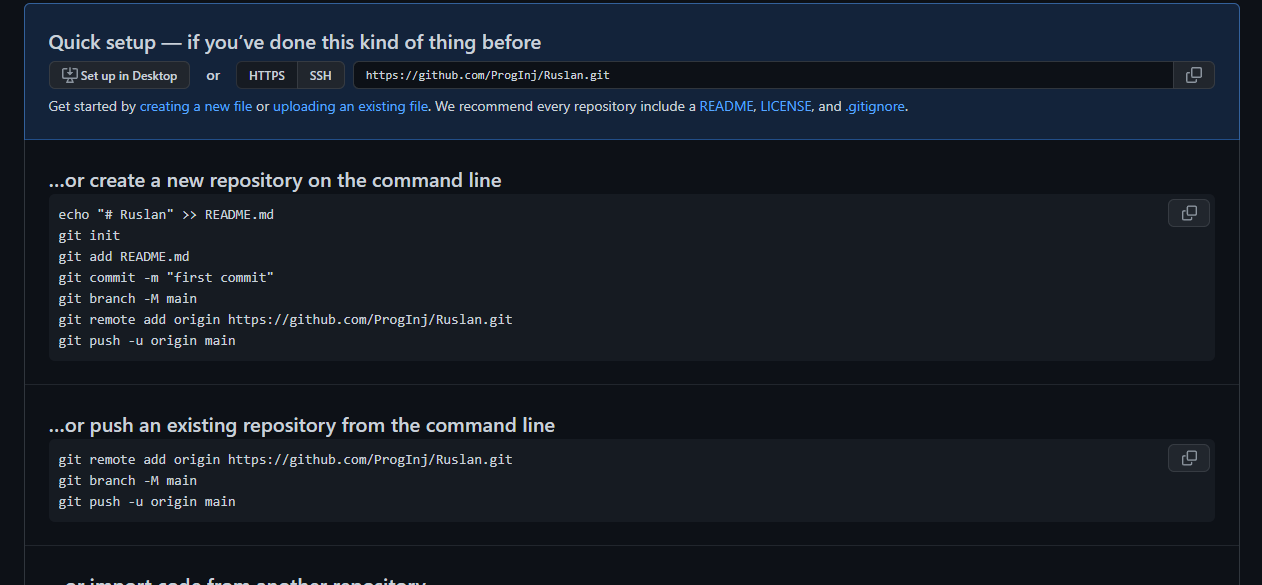
1. Освоить передачу истории хранилища по сети.
   1. Организовать общее хранилище на удаленном сервере.
      1. Зарегистрироваться на [GitHub.](https://github.com/)



* + 1. Создать пустое удаленное хранилище с любым наименованием.



Создали пустое хранилище:

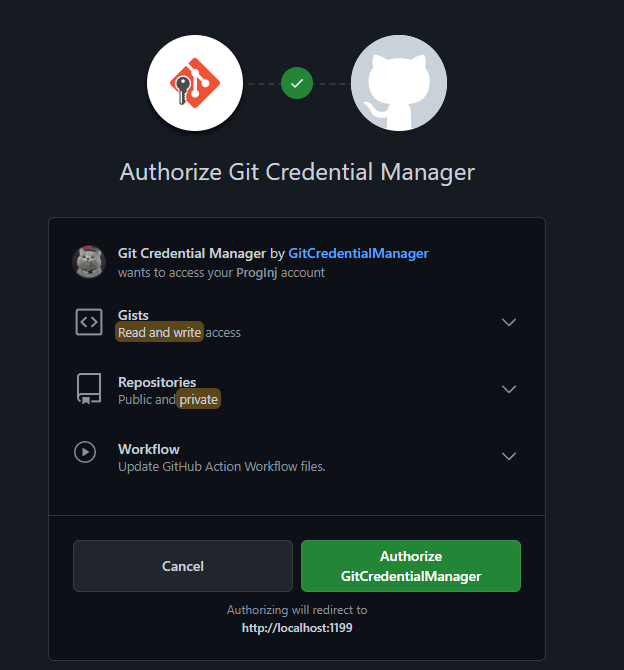


* + 1. Настроить локальное хранилище для синхронизации с удаленным. **От загрузки сведений о ветвлениях в удаленном хранилище отказаться.**

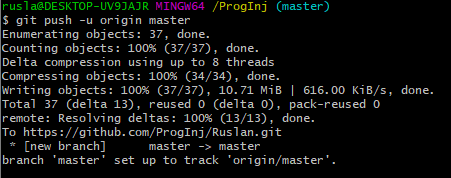
Вводим команду «git remote add origin [https://github.com/ProgInj/Ruslan.git»](https://github.com/ProgInj/Ruslan.git). Данная команда добавит удаленный репозиторий.



Далее вводим команду «git push» чтоб отправить наши данные репозитория на github, при отправке запросит ввести данные аккаунта от github и авторизоваться в нем.



1. Передать локальное хранилище на удаленный сервер (push). После ввода авторизации начнется процесс передачи файлов.



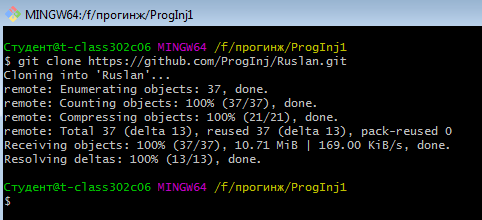
1. Перейти к странице хранилища на GitHub (обновить её) и ознакомиться с возможностями просмотра содержимого через web-интерфейс.



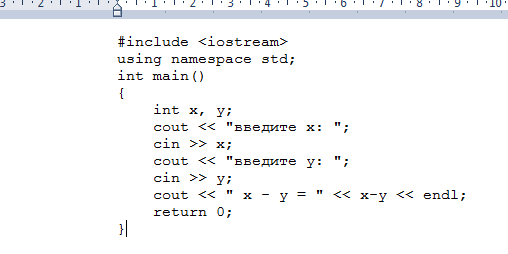
1. Загрузить копию удаленного хранилища на локальную машину (clone).

Целью является имитация совместной работы с удаленным хранилищем.Для этого на одной машине организуются 2 локальных хранилища: созданное в пункте [1.1](#page1) (RepoA) и загруженное с удаленного сервера (RepoB).

Диалог«Git clone»следует вызывать из контекстного меню каталога *вне* локального хранилища.В качестве *URL* потребуется указать адрес удаленногохранилища, а в качестве *Directory* — имя каталога для нового локального хранилища.

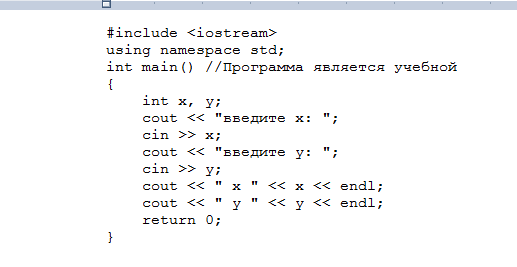


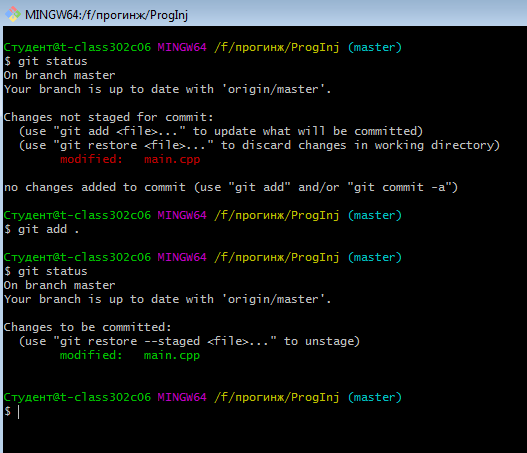
* 1. В локальном хранилище RepoB добавить в программу печать разности введенных чисел, сделать commit и передать изменения на сервер.



ЗАГРУЗИТЬ НА ХАБ

* 1. В локальном хранилище RepoA добавить над функцией main() комментарий о том, что программа является учебной, сделать commit, **но не отправлять изменений на сервер.**





1. На странице хранилища на GitHub перейти в раздел *Commits* и ознакомиться с возможностью просмотра истории изменений через web-интерфейс.
2. В локальном хранилище RepoA выполнить загрузку с сервера новейших ветвлений и изменений (fetch) и просмотреть журнал хранилища.
3. Совместить изменения в локальном хранилище с загруженными.
   1. Использовать действие pull для загрузки изменений с удаленного сервера и автоматического совмещения их с имеющимися локально. Просмотреть журнал изменений.

2.8.2. Отменить неудобный результат действия pull.

1. Выполнить перенос (rebase) локальных изменений на основу новейшего загруженного состояния проекта.
2. Передать итоговое состояние локального хранилища RepoA на удаленный сервер, используя команду push.
3. На данном этапе во всех трех хранилищах(локальныхRepoAиRepoB и удаленном на GitHub) должна быть одинаковая линейная история из пяти — шести commit-ов.
4. Изучить действия, связанные с ветвлениями и разрешением конфликтов.
   1. Добавить в программу печать произведения чисел и совершите commit.

#include "sdt.h"

// This program is just an example one under VCS.

int main()

{

int a, b;

cout << "Enter A and B: ";

cin >> a >> b;

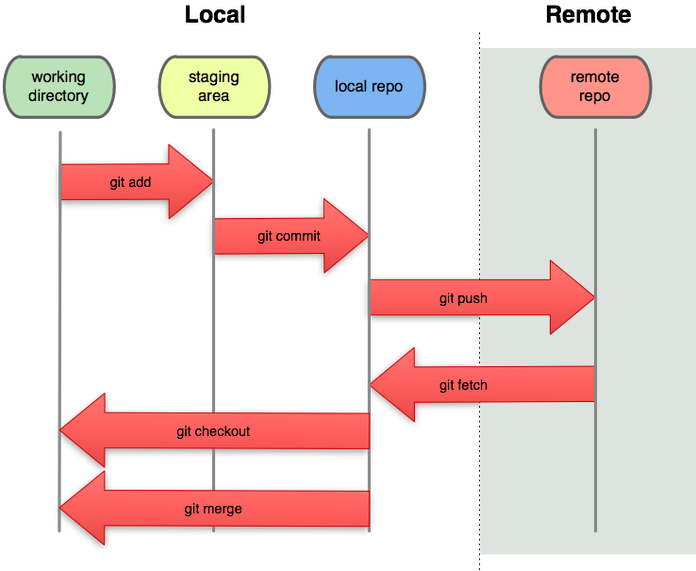
cout << "A + B = " << a + b << '\n'

cout << "A - B = " << a - b << '\n'

cout << "A \* B = " << a \* b << '\n';

}

1. В новой ветви повторить пункт [3.1,](#page4) заменив умножение делением.



3.5.2. Приступить к разрешению конфликта.

В нижнем поле каждый восклицательный знак обозначает отдельный конфликт.

Примерный вид окна редактирования конфликтов представлен на рисунке ниже.

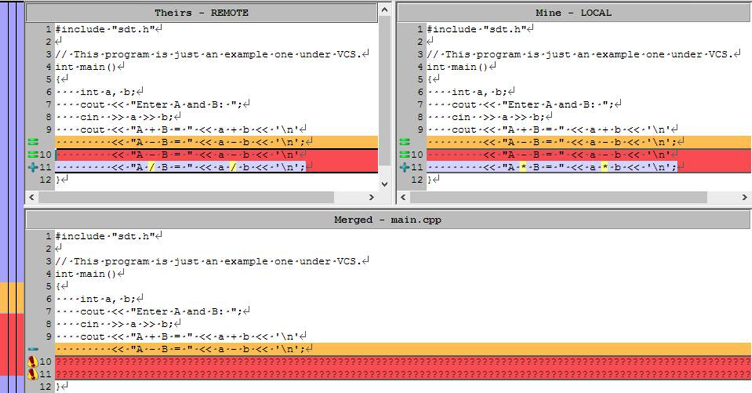


Рис.1 – Вид окна редактирования конфликтов

1. Разрешить конфликты:

— в качестве строки 10 предпочесть строку 10 любой ветви;

— на место строки 11 вставить 2 строки: строку с печатью произведения и строку с печатью частного;

— лишнюю точку с запятой на строке 11 поля-результата удалить.

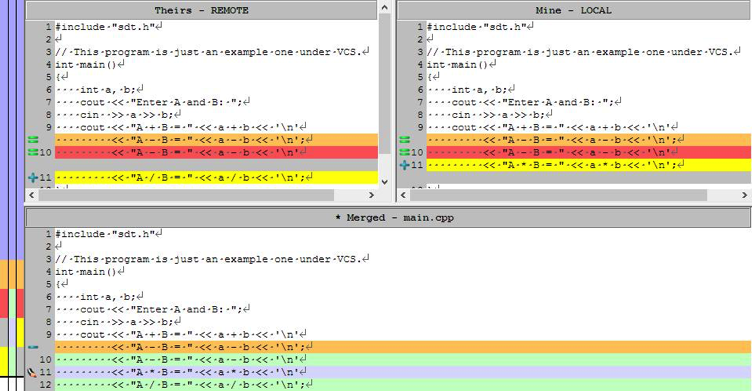


Рис.2 – Вид окна редактирования конфликтов при слиянии правок

3.5.4. Завершить процедуру разрешения конфликтов.

1. Завершить слияние ветви division в ветвь master, написав осмысленный комментарий к слиянию и совершив commit.
2. Убедиться, что программа компилируется и верно работает. Если это не так, исправить все ошибки и добиться правильной работы. Совершить commit.

Вывод: Проделав лабораторную работу, были изучены на практике понятия и компоненты систем контроля версий (СКВ), приемы работы с ними и освоено специализированное ПО и распространенный сервис для работы с распределенной СКВ Git — git-bash и GitHub.com.