Министерство науки и высшего образования

Российской федерации

Федеральное государственное бюджетное

Образовательное учреждение высшего образования

«Новосибирский Государственный Технический Университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа №5

«Системы контроля версий»

Факультет: прикладной информатики и информатики

Группа: ПМИ-12  
Бригада: 1

Студенты: Михайловский М.А.

Швадченко А.В.

Преподаватели: Кобылянский В.Г.

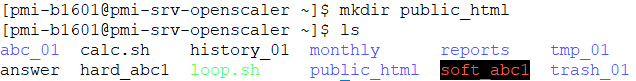
Филиппова Е.В.

Новосибирск, 2023

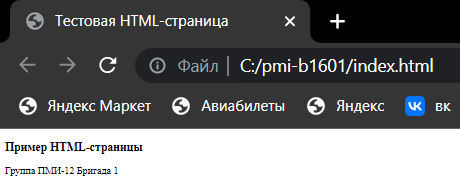
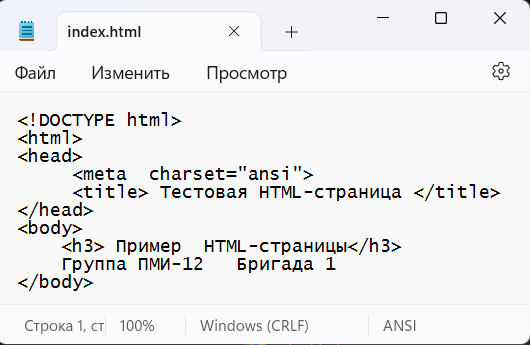
1. Цель работы:
2. Целью выполнения работы является изучение систем контроля версий git и GitLab, а также подготовка репозитория для выполнения лабораторной работы № 6.
3. Выполнение работы:

Часть 1

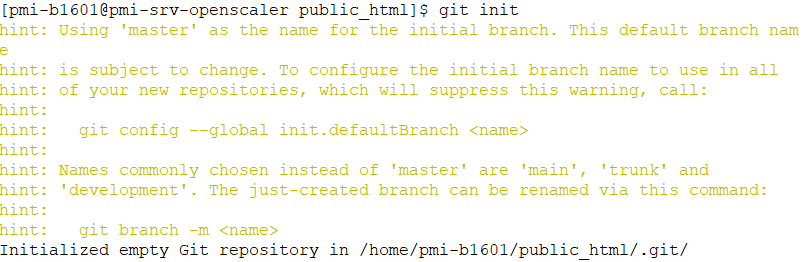
1. В домашнем каталоге Open Scaler создать каталог **public\_html**.



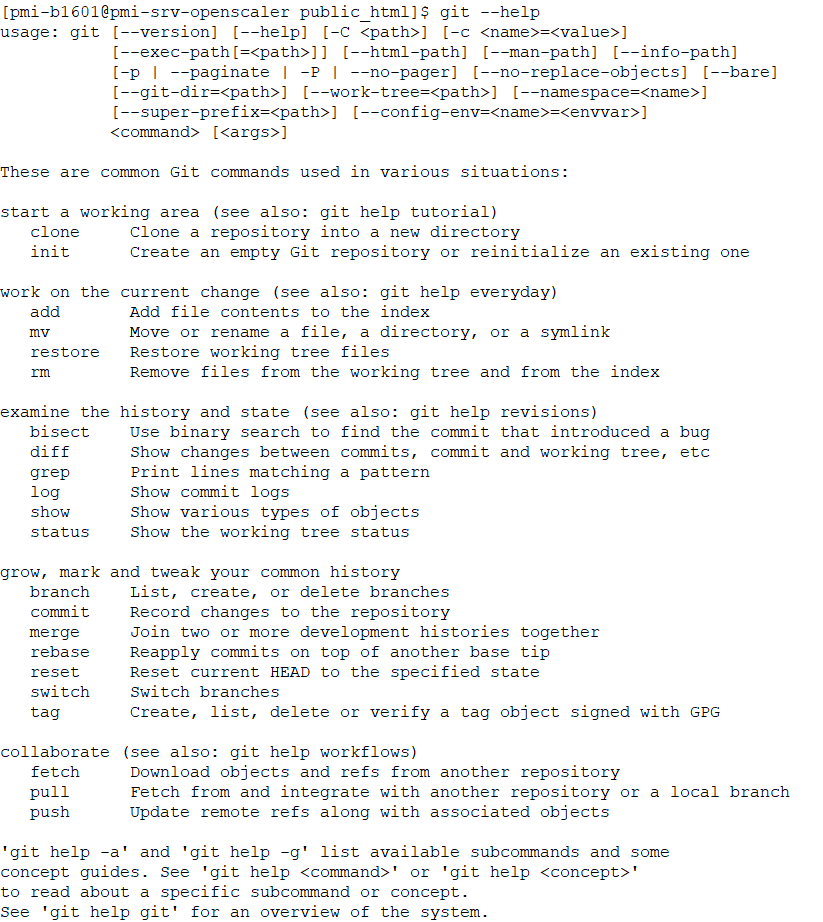
1. В Блокноте Windows создать текстовый файл в кодировке ANSI с именем **index.html**, текст которого приведен в разделе 2.4. Проверить работу этого файла и при отсутствии ошибок скопировать его в каталог **public\_html** сервера.



1. Перейти в каталог **public\_html** сервера и создать в нем локальный git-репозиторий.



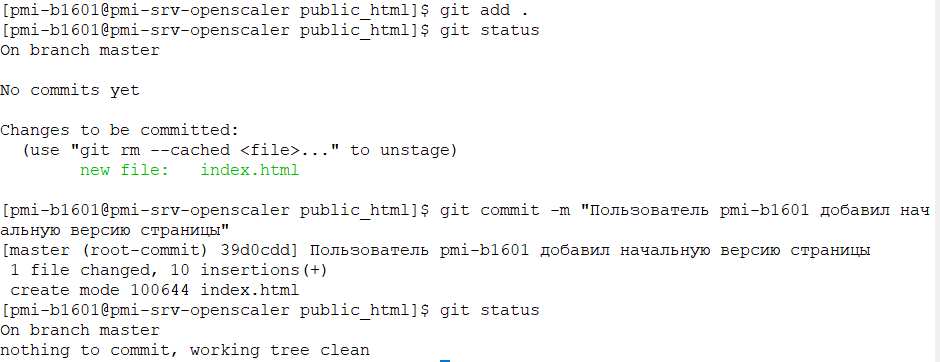
1. Вывести список команд git.



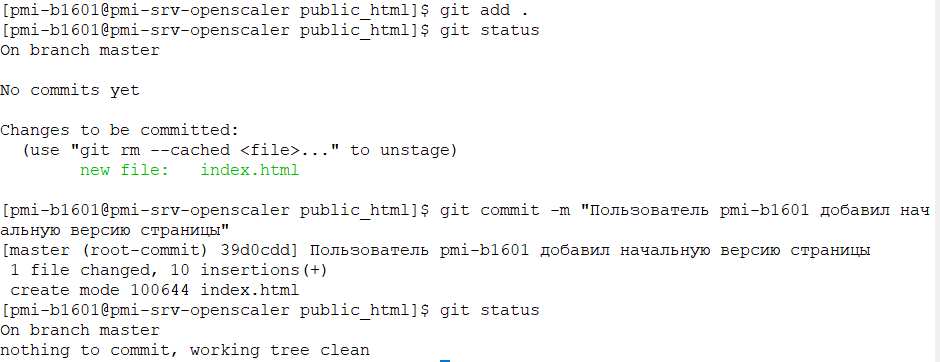
1. Выполнить глобальные настройки репозитория путем установки имени пользователя и адреса его электронной почты.



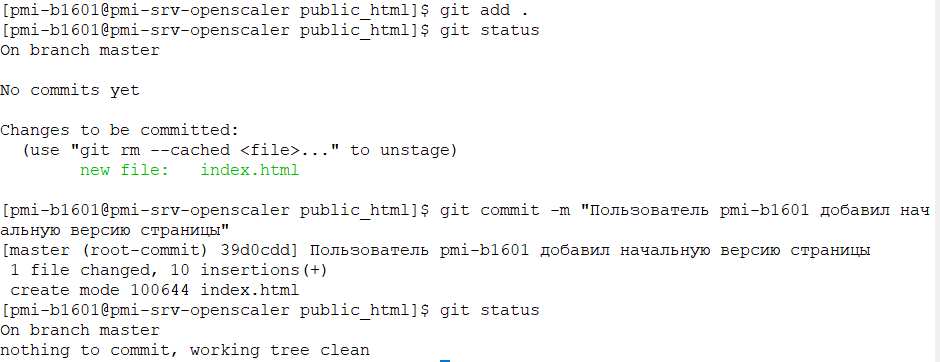
1. Добавить файл **index.html** в область индексирования репозитория.



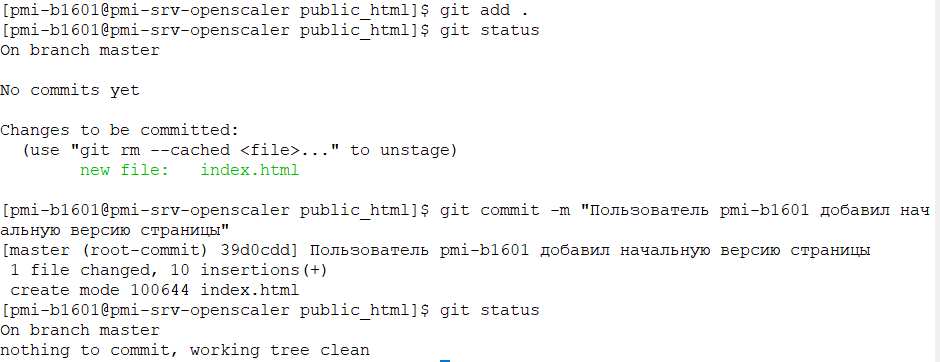
1. Определить состояние репозитория командой *status.*



1. Выполнить начальный коммит с добавлением комментария, например «Пользователь pmi-b1105 добавил начальную версию страницы»

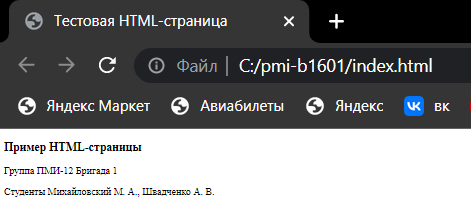
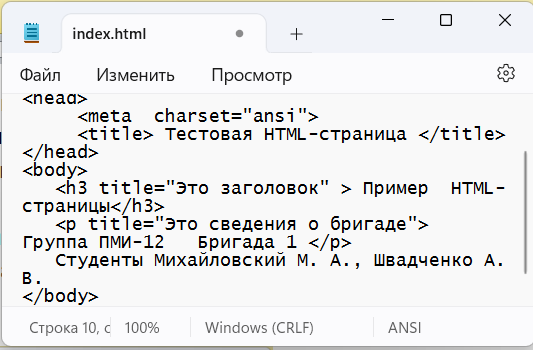


1. Повторно определить состояние репозитория, сравнить результаты с п. 7.

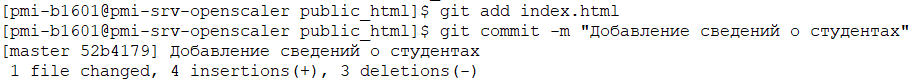


1. Выполнить все задания раздела 2.4 по редактированию файла **index.html.** Редактирование рекомендуетсяпроводить в среде Windows. После каждого внесения изменений необходимо после проверки корректности работы файла скопировать его в каталог **public\_html** сервера и сохранить обновленную версию в репозитории с соответствующим комментарием.

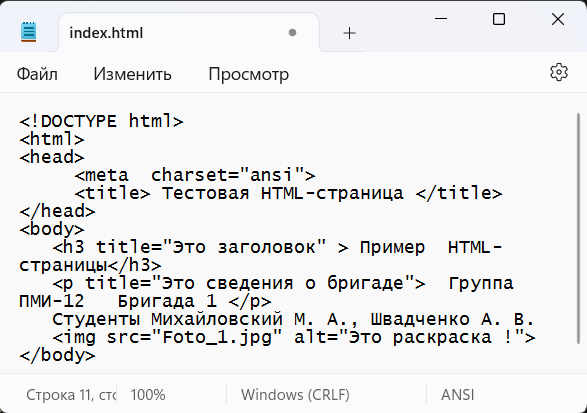
Вносим изменения в файл **index.html**: группу и номер бригады оформляем в виде абзаца, добавляем сведения о студентах бригады и всплывающие подсказки.

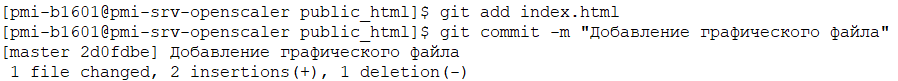


Добавляем измененный файл в индекс и выполняем коммит

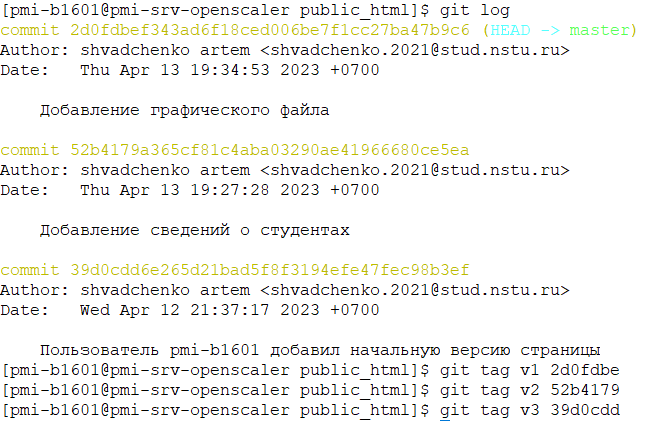


Изменяем файл **index.html (**добавляем любой графический файл) и выполняем коммит

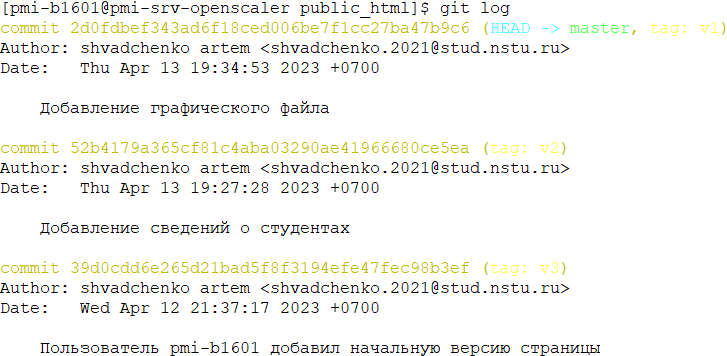




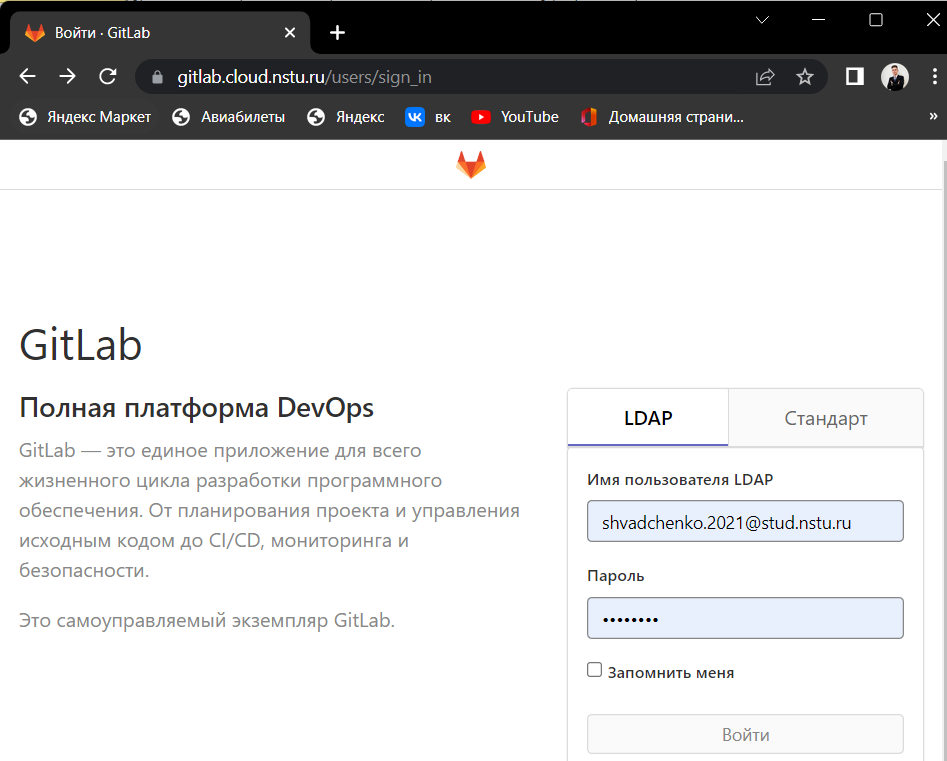
1. Создать отдельный тэг для каждого коммита.



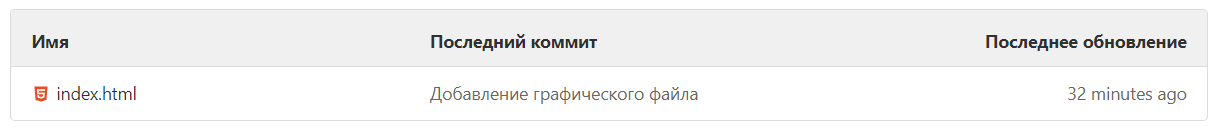
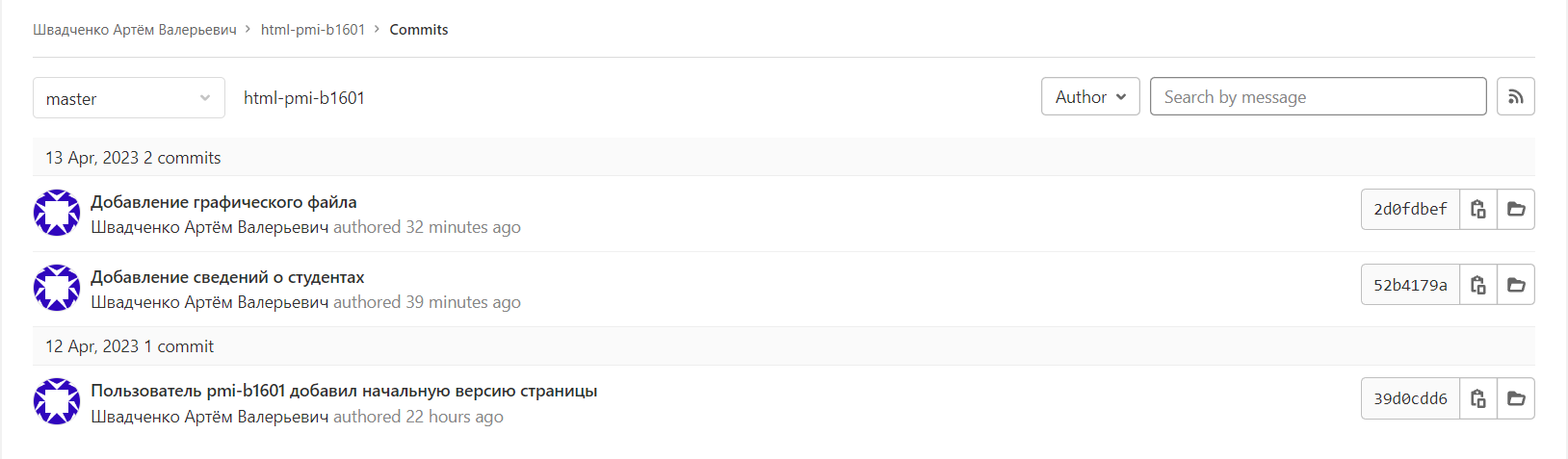
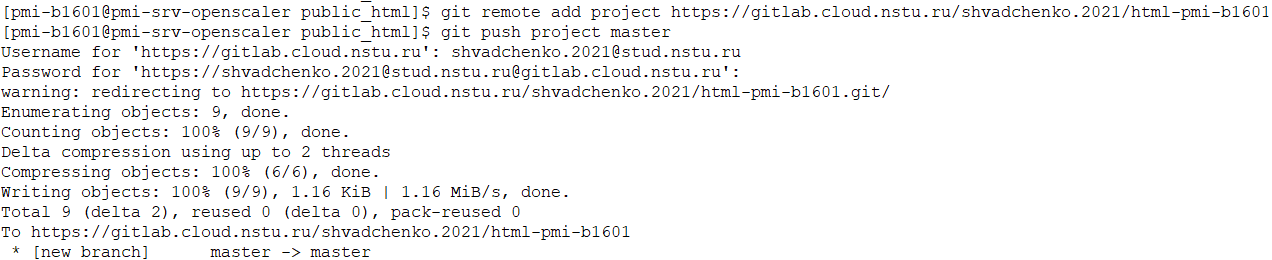
1. Вывести журнал всех изменений файла **index.html.**



1. Войти в систему GitLab на облачной платформе НГТУ путем ввода логина и пароля от личного кабинета.



1. Создать проект, заполнив соответствующую форму (см. рис.12). Имя проекта должно содержать номер Вашей группы и бригады, например, *html-pmi-b1105,* уровень видимости проекта – *internal.*
2. Загрузить все данные локального репозитория каталога **public\_html** в проект, созданный в п. 14.



Часть 2

1. В домашнем каталоге сервера Open Scaler создать рабочий каталог для разработки и отладки программ, его имя должно содержать бригадный логин, например, **git-pmi-b1105**.





1. Проверить работоспособность программ, разработанных по курсам «Структуры данных и алгоритмы» (для групп ПМ) или « Информационные технологии и основы программирования» (для групп ПМИ) в среде Windows. У каждого студента должен быть свой вариант программы (для бригады из двух человек далее – **prog1** и **prog2**).  
   prog1 – Швадченко А. В.

Дана непустая последовательность чисел (целых) используя структуры данных определить сумму последовательности

prog2 – Михайловский М. А.

Дана непустая последовательность чисел (целых) используя структуры данных определить среднее арифметическое последовательности

Содержание программы prog1.c:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

typedef struct sNode {

int value;

struct sNode\* next;

}list;

list\* read\_list() {

list\* head = malloc(sizeof(list));

head->next = NULL;

list\* p = head;

int val;

FILE\* file;

int u;

printf("Enter the test file number");

scanf("%d", &u);

char\* file\_name = "";

if (u == 1)

{

file\_name = "elems1.txt\0";

}

else if (u == 2)

{

file\_name = "elems2.txt\0";

}

else if (u == 3) {

file\_name = "elems3.txt\0";

}

fopen\_s(&file, file\_name, "r");

while (fscanf\_s(file, "%d", &val) != EOF) {

p->next = malloc(sizeof(list));

p = p->next;

p->value = val;

p->next = NULL;

}

fclose(file);

return head;

}

list\* enter\_list() {

list\* head = malloc(sizeof(list));

list\* p = head;

int n;

int value;

printf\_s("Enter the number of totalized objects:");

scanf("%d", &n);

while (n > 0) {

scanf("%d", &value);

p->next = malloc(sizeof(list));

p = p->next;

p->value = value;

p->next = NULL;

n--;

}

return head;

}

int sum\_list(list\* head) {

int result = 0;

list\* p = head->next;

while (p != NULL) {

result += p->value;

p = p->next;

}

return result;

}

int main() {

int option;

printf("This program calculates the sum of the list items. the list is filled in from a file or from the keyboard with integer values. The operating mode is selected in the following menu");

printf("1 - Read data from file\n2 - Enter data on keyboard\nEnter a number for the operation:");

scanf("%d", &option);

list\* lst;

if (option == 1) {

lst = read\_list();

}

else if (option == 2) {

lst = enter\_list();

}

else {

printf("No such operation!\n");

lst = malloc(sizeof(list));

lst->next = NULL;

}

int sum = sum\_list(lst);

printf("Total: %d", sum);

}

Содержание программы prog2.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

typedef struct sNode {

int value;

struct sNode\* next;

}list;

list\* read\_list() {

list\* head = malloc(sizeof(list));

head->next = NULL;

list\* p = head;

int val;

FILE\* file;

int u;

printf("Enter the test file number");

scanf("%d", &u);

char\* file\_name = "";

if (u == 1)

{

file\_name = "elems1.txt\0";

}

else if (u == 2)

{

file\_name = "elems2.txt\0";

}

else if (u == 3) {

file\_name = "elems3.txt\0";

}

fopen(&file, file\_name, "r");

while (fscanf\_s(file, "%d", &val) != EOF) {

p->next = malloc(sizeof(list));

p = p->next;

p->value = val;

p->next = NULL;

}

fclose(file);

return head;

}

list\* enter\_list() {

list\* head = malloc(sizeof(list));

list\* p = head;

int n;

int value;

printf("Enter the number of totalized objects:");

scanf("%d", &n);

while (n > 0) {

scanf("%d", &value);

p->next = malloc(sizeof(list));

p = p->next;

p->value = value;

p->next = NULL;

n--;

}

return head;

}

float average\_list(list\* head) {

float sum = 0;

float count = 0;

list\* p = head->next;

while (p != NULL) {

sum += p->value;

count += 1;

p = p->next;

}

float result = sum / count;

return result;

}

int main() {

int option;

printf("1 - Read data from file\n2 - Enter data on keyboard\nEnter a number for the operation:");

scanf("%d", &option);

list\* lst;

if (option == 1) {

lst = read\_list();

}

else if (option == 2) {

lst = enter\_list();

}

else {

printf("No such operation!\n");

lst = malloc(sizeof(list));

lst->next = NULL;

}

float sum = average\_list(lst);

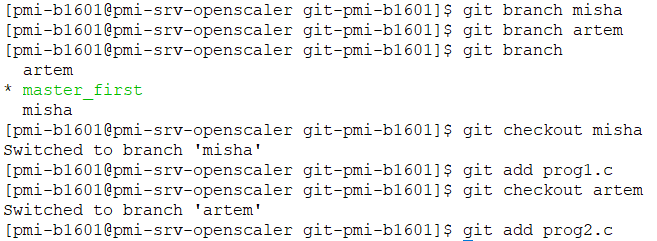
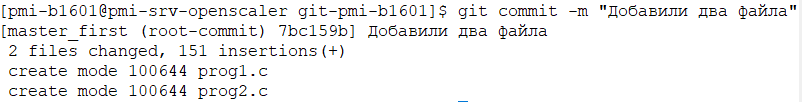
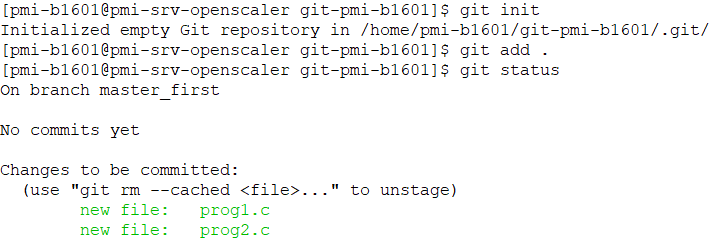
printf("Total: %f", sum);

}

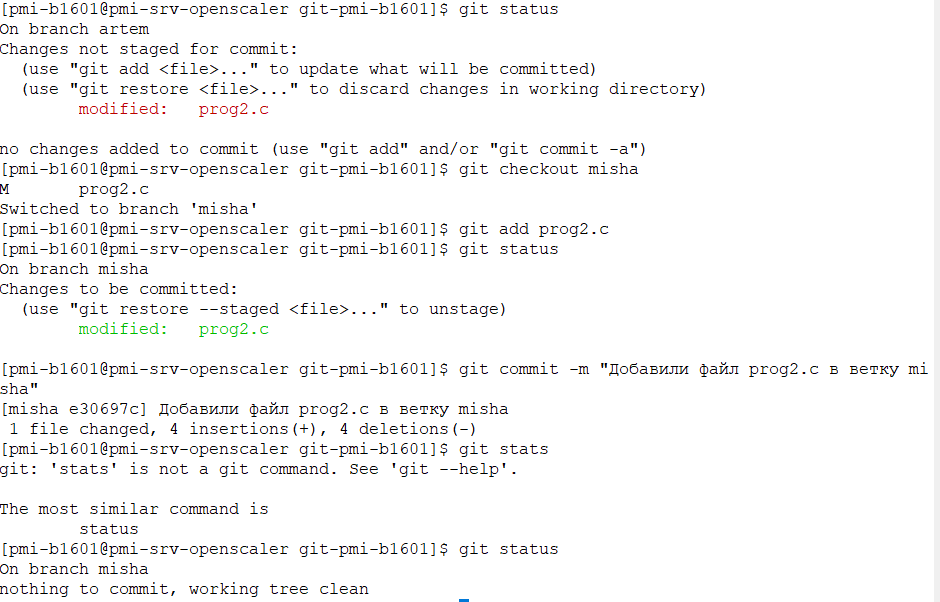
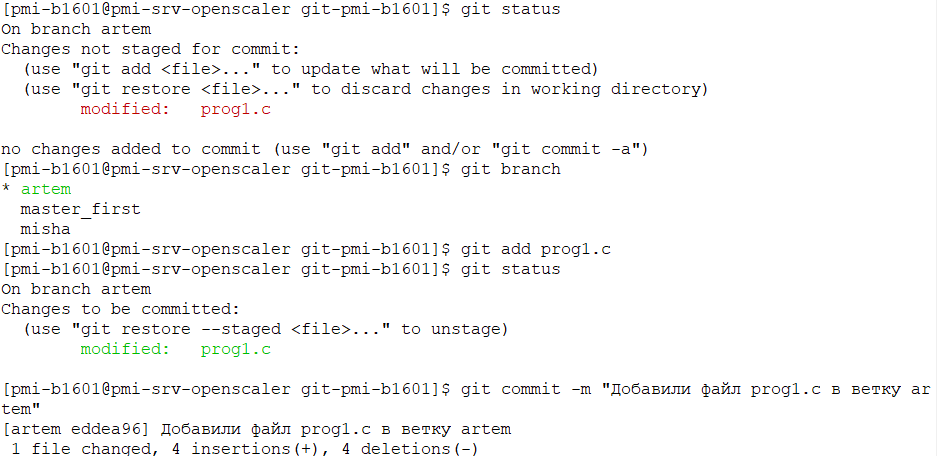
1. Скопировать файлы с исходными текстами программ в созданный рабочий каталог с помощью WinSCP.



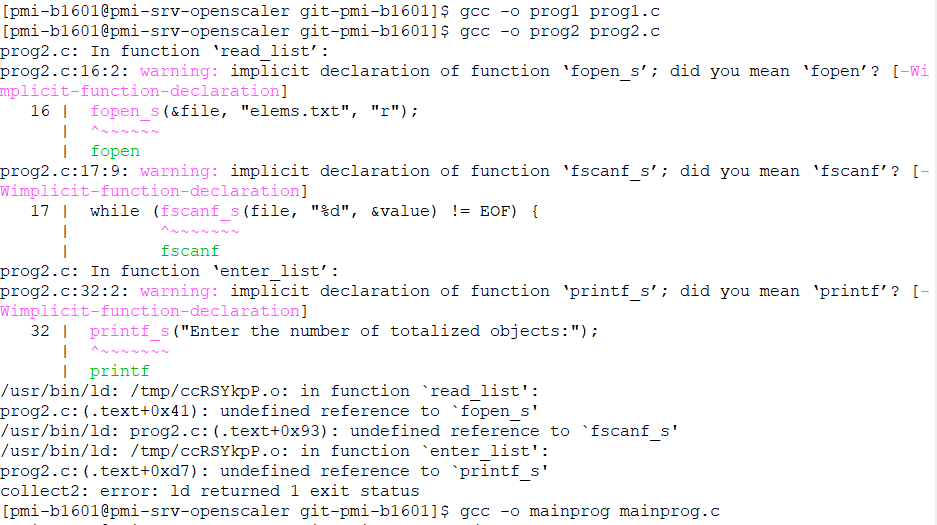
1. Перейти в рабочий каталог, создать в нем новый git-репозиторий и добавить в него исходные файлы проекта. Файлы **prog1.c** и **prog2.c** должны быть сохранены в разные ветки репозитория.







1. Доработать тексты программы с учетом требований

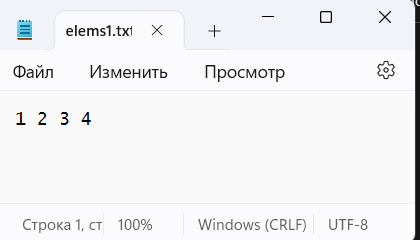
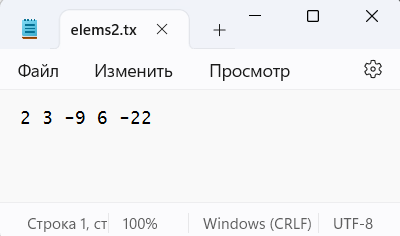


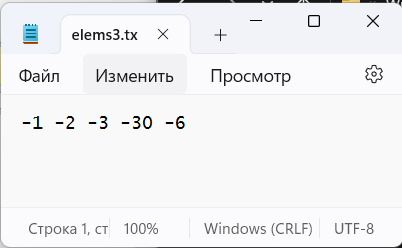




1. Разработать файлы с тестовыми наборами данных (не менее трех).

В тестовом файле elems1.txt содержатся целые положительные числа, в файле elems2.txt содержатся положительные и отрицательные числа, в файле elems3.txt содержатся отрицательные числа.



1. В среде Windows разработать новую программу **mainprog**, главная функция которой формирует меню, состоящее из трех пунктов: вызов **prog1,** вызов **prog2** и завершение программы. Для этого код каждой из программ надо оформить в виде отдельной функции.

Текст программы mainprog.c:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

typedef struct sNode {

int value;

struct sNode\* next;

}list;

list\* read\_list() {

list\* head = malloc(sizeof(list));

head->next = NULL;

list\* p = head;

int value;

FILE\* file;

int u;

printf("Enter the test file number");

scanf("%d", &u);

char\* file\_name = "";

if (u == 1)

{

file\_name = "elems1.txt\0";

}

else if (u == 2)

{

file\_name = "elems2.txt\0";

}

else if (u == 3) {

file\_name = "elems3.txt\0";

}

file = fopen(file\_name, "r");

while (fscanf\_s(file, "%d", &value) != EOF) {

p->next = malloc(sizeof(list));

p = p->next;

p->value = value;

p->next = NULL;

}

fclose(file);

return head;

}

list\* enter\_list() {

list\* head = malloc(sizeof(list));

list\* p = head;

int n;

int value;

printf\_s("Enter the number of totalized objects:");

scanf("%d", &n);

while (n > 0) {

scanf("%d", &value);

p->next = malloc(sizeof(list));

p = p->next;

p->value = value;

p->next = NULL;

n--;

}

return head;

}

int sum\_list(list\* head) {

int result = 0;

list\* p = head->next;

while (p != NULL) {

result += p->value;

p = p->next;

}

return result;

}

float average\_list(list\* head) {

float sum = 0;

float count = 0;

list\* p = head->next;

while (p != NULL) {

sum += p->value;

count += 1;

p = p->next;

}

float result = sum / count;

return result;

}

int sum() {

int option;

printf("1 - Read data from file\n2 - Enter data on keyboard\nEnter a number for the operation:");

scanf("%d", &option);

list\* lst;

if (option == 1) {

lst = read\_list();

}

else if (option == 2) {

lst = enter\_list();

}

else {

printf("No such operation!\n");

lst = malloc(sizeof(list));

lst->next = NULL;

}

int sum = sum\_list(lst);

return sum;

}

float average() {

int option;

printf("1 - Read data from file\n2 - Enter data on keyboard\nEnter a number for the operation:");

scanf("%d", &option);

list\* lst;

if (option == 1) {

lst = read\_list();

}

else if (option == 2) {

lst = enter\_list();

}

else {

printf("No such operation!\n");

lst = malloc(sizeof(list));

lst->next = NULL;

}

float avrg = average\_list(lst);

return avrg;

}

int main() {

int option;

int result1;

float result2;

printf("1 - prog1\n2 - prog2\nEnter a number for the program:");

scanf("%d", &option);

if (option == 1) {

result1 = sum();

printf("Result of prog1: %i", result1);

}

else if (option == 2) {

result2 = average();

printf("Result of prog2: %f", result2);

}

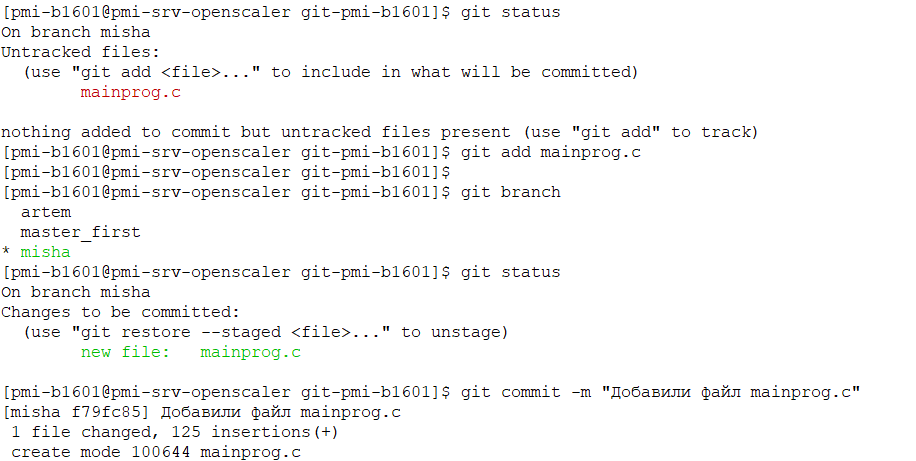
else {

printf("No such function!\n");

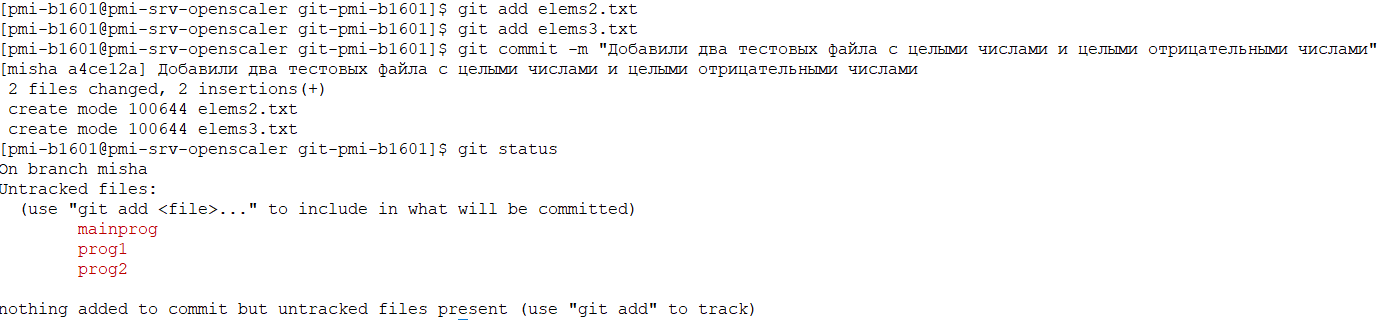
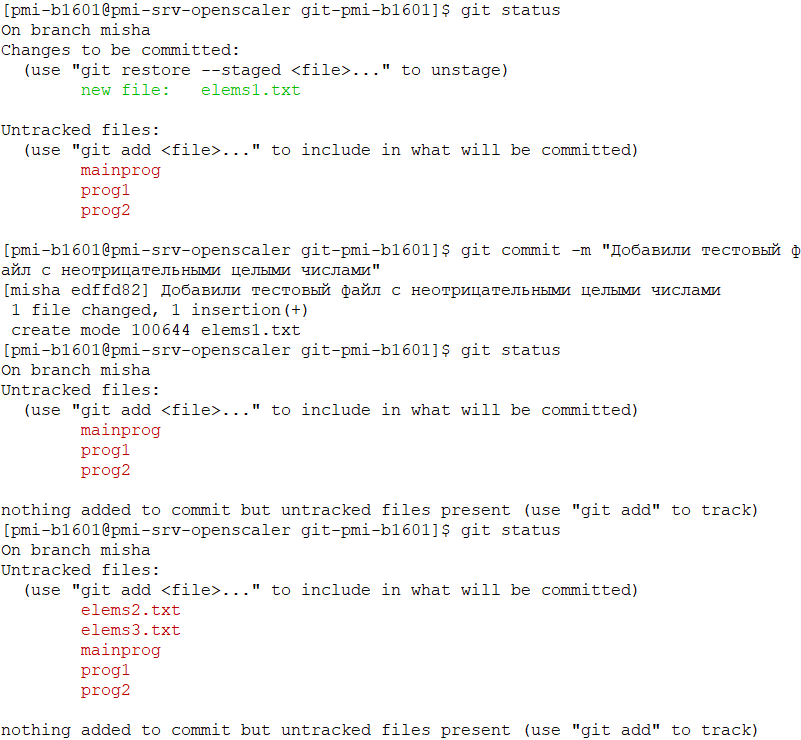
}

}

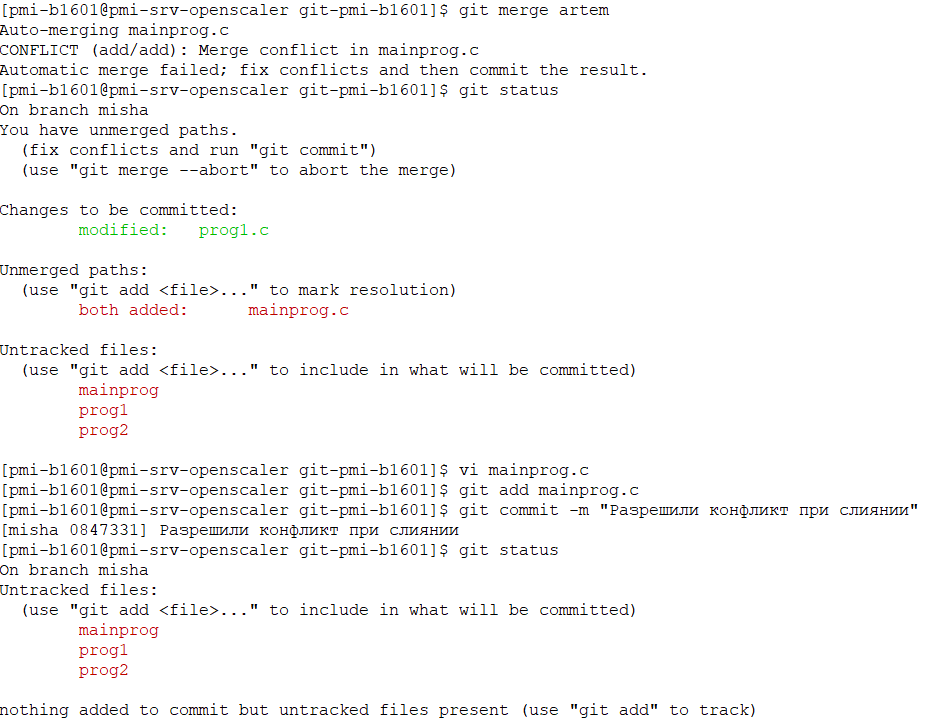
1. Скопировать файл с исходным текстом программы **mainprog.c** в рабочий каталог и сохранить его в одной из веток репозитория.



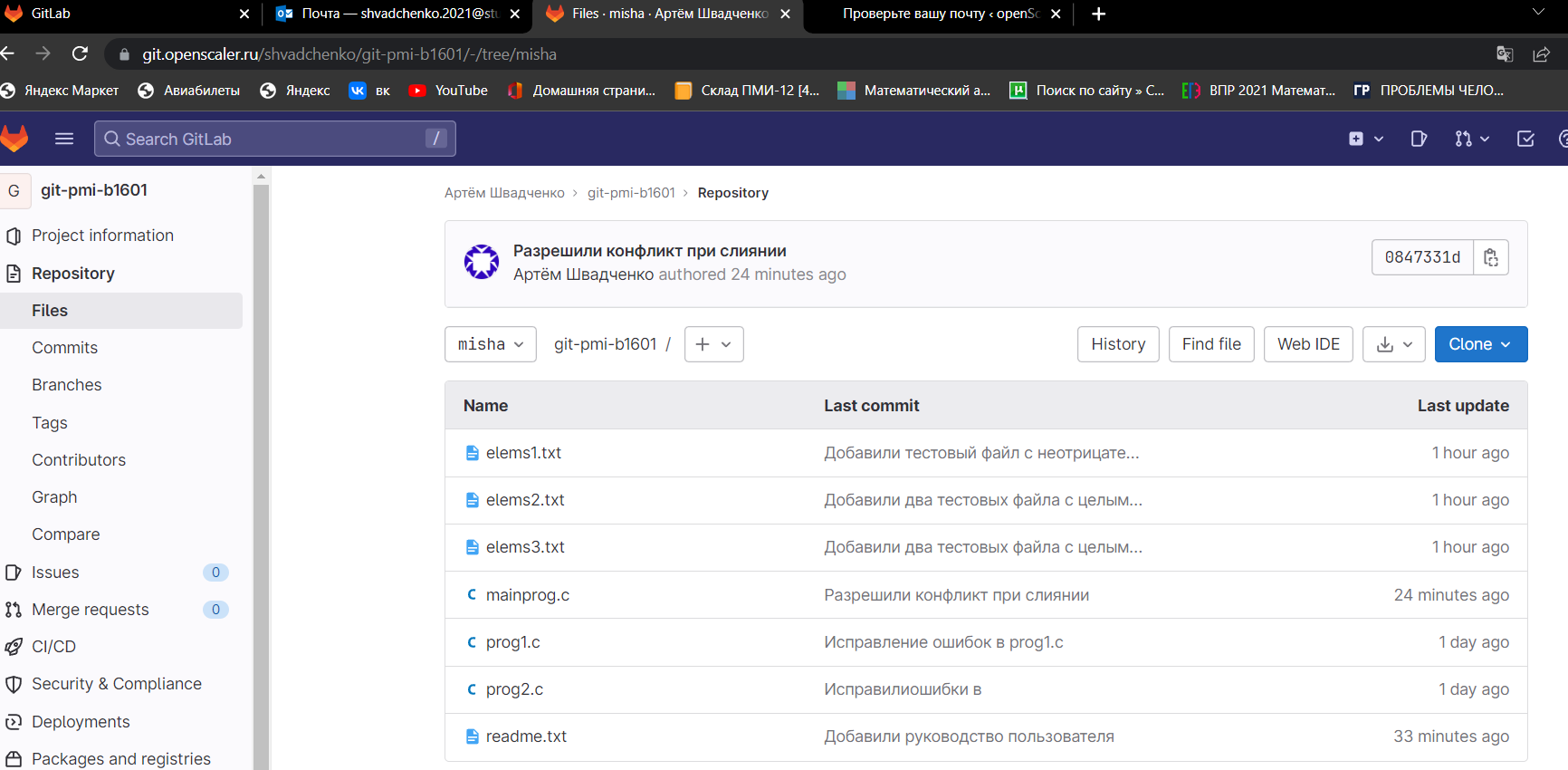
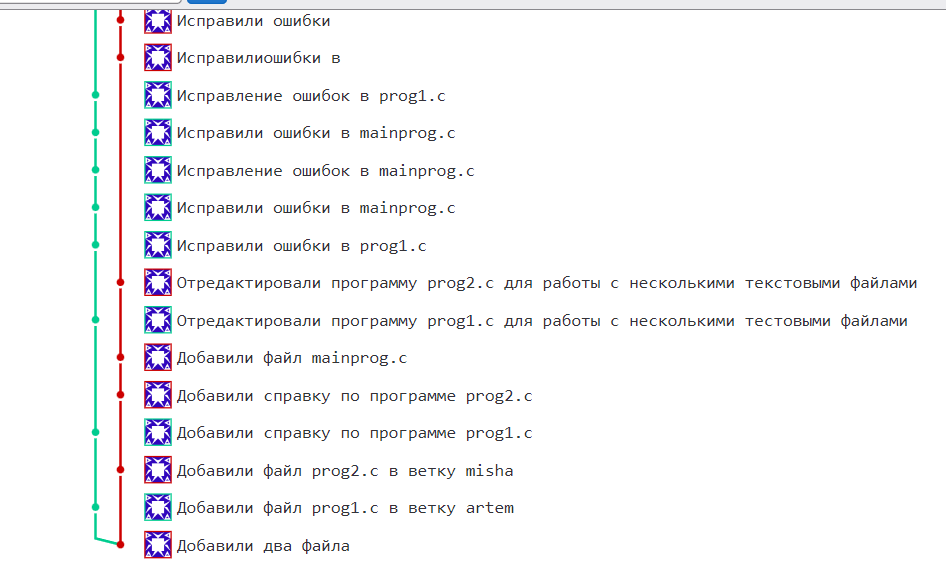
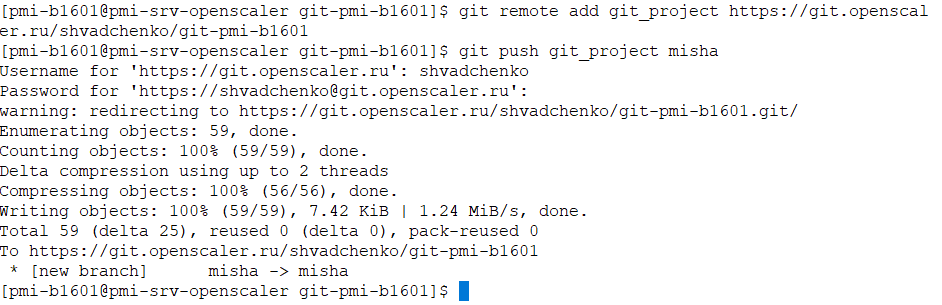
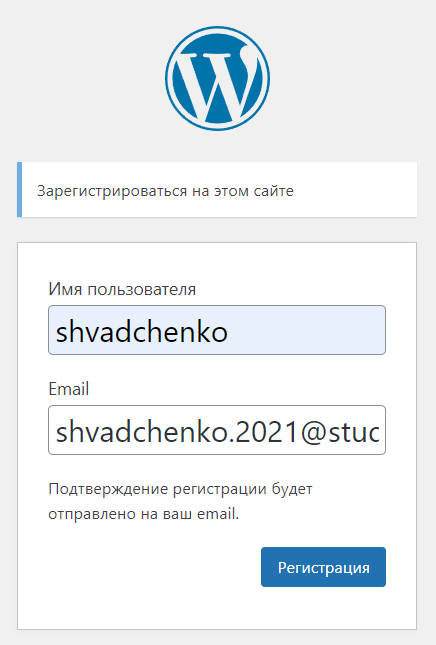
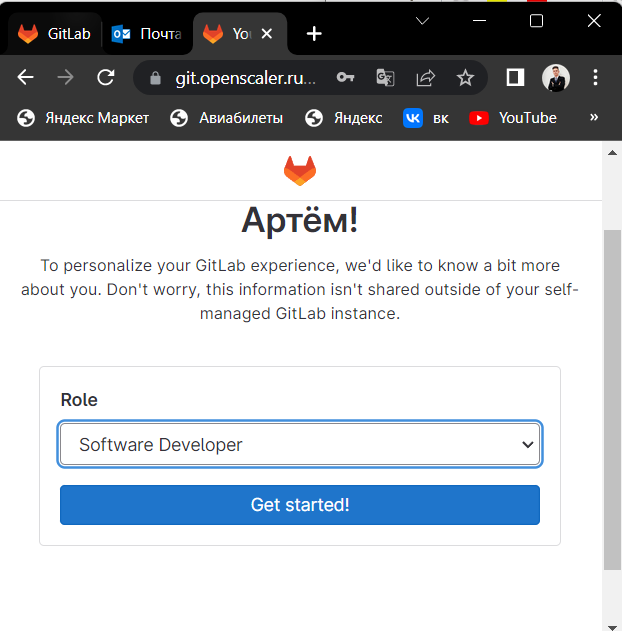
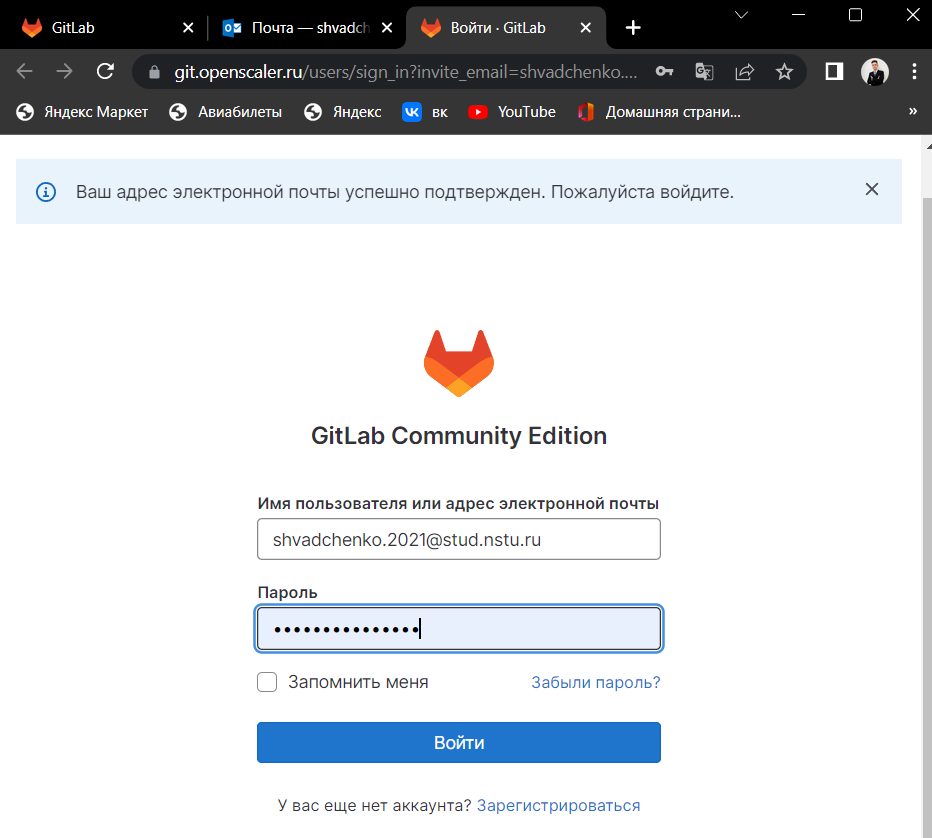
1. Сохранить в репозитории файлы с руководством пользователя и тестовыми наборами данных.



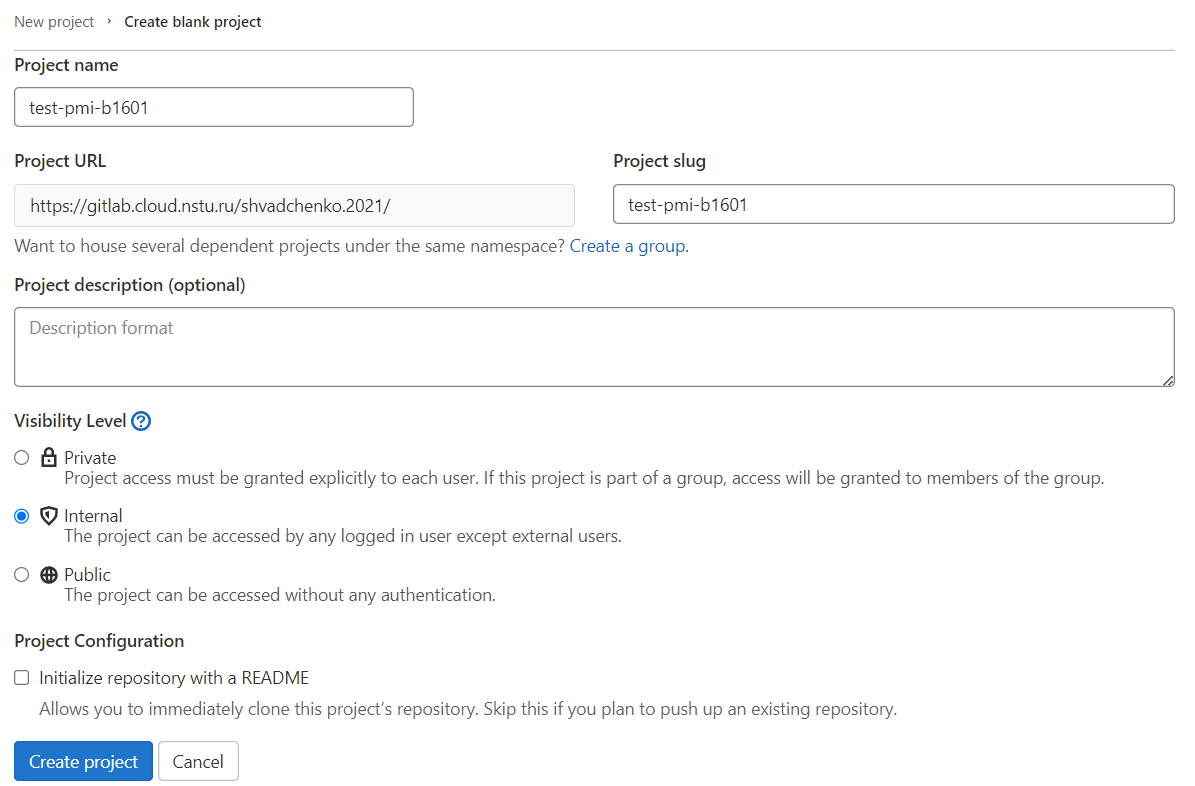
1. Выполнить слияние веток репозитория.

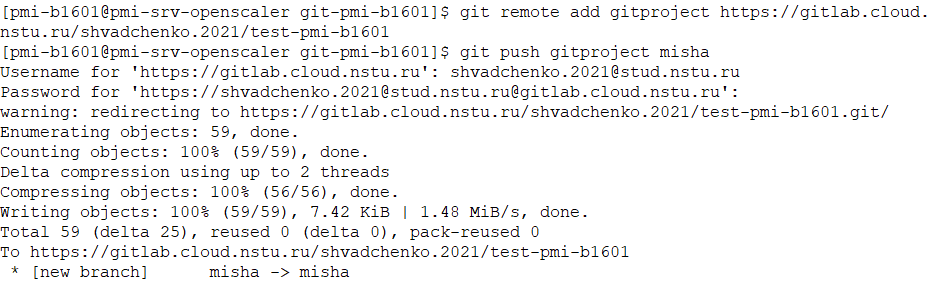


1. Зарегистрироваться в централизованном репозитории сообщества *git.openscaler.ru* и загрузить в него все файлы из локального репозитория. Имя проекта уточнить у преподавателя.



1. При работе в системе GitLab облачной платформы НГТУ создать новый проект *test-pmi-b1105* и загрузить в него все файлы из локального репозитория.





1. Изучили системы контроля версий. Проработали контрольные вопросы по теме.