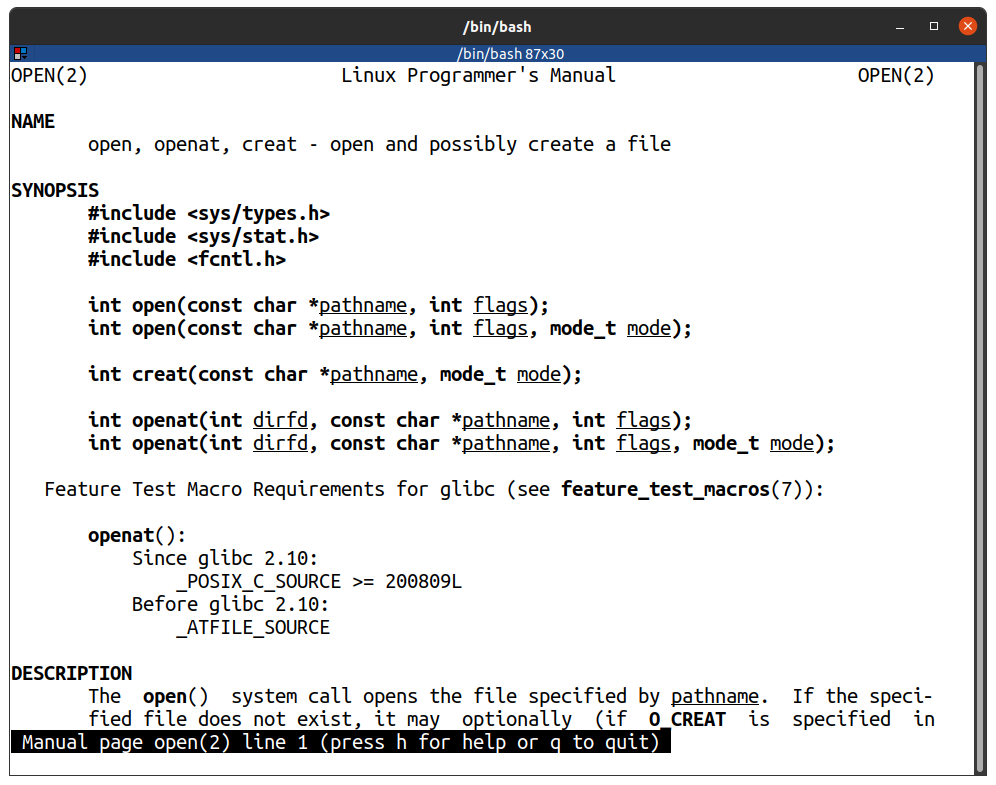
1. Изучить порядок работы с системной справкой man в ОС семейства Linux.
2. Освоить последовательность действий по созданию программ для ОС Linux в среде CLion.
3. Освоить последовательность действий по отладке программ в среде разработки CLion.
4. Разработать приложение, реализующее безопасное чтение строк из стандартного потока ввода с их последующей записью в файл. Имя файла и размер принимаемого буфера задаются в качестве входных параметров при запуске приложения.
5. Модифицировать разработанное приложение таким образом, чтобы добавление строк в файл осуществлялось в обратном порядке относительно их прихода.

# Порядок работы с системной справкой man в ОС семейства Linux

Команда man позволяет получить доступ к общей базе справки по команде, функции или программе. Для просмотра справки программе надо передать название команды или другого объекта в системе.

Рисунок 1.1 – Результат выполнения команды man open

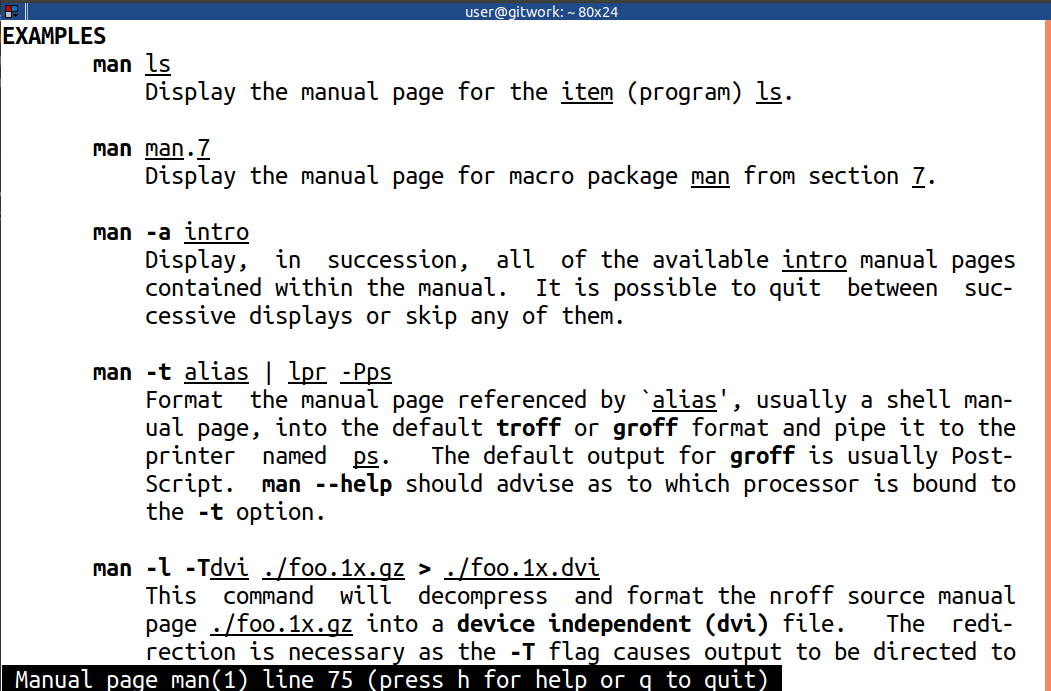


Рисунок 1.2 – Результат выполнения команды man man

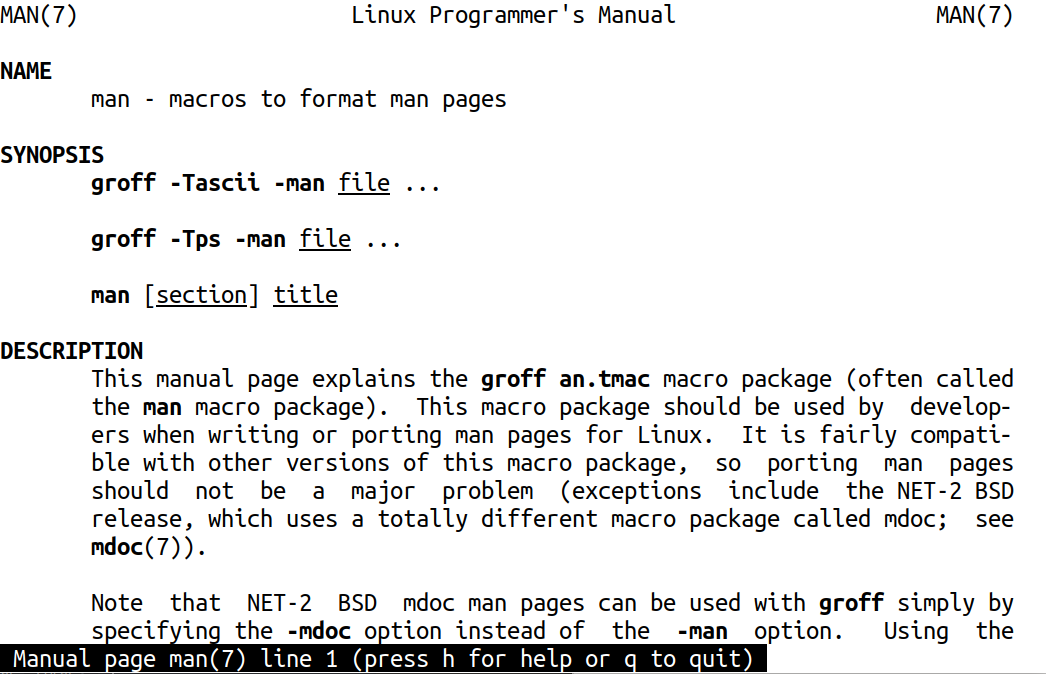


Рисунок 1.3 – Результат выполнения команды man man.7

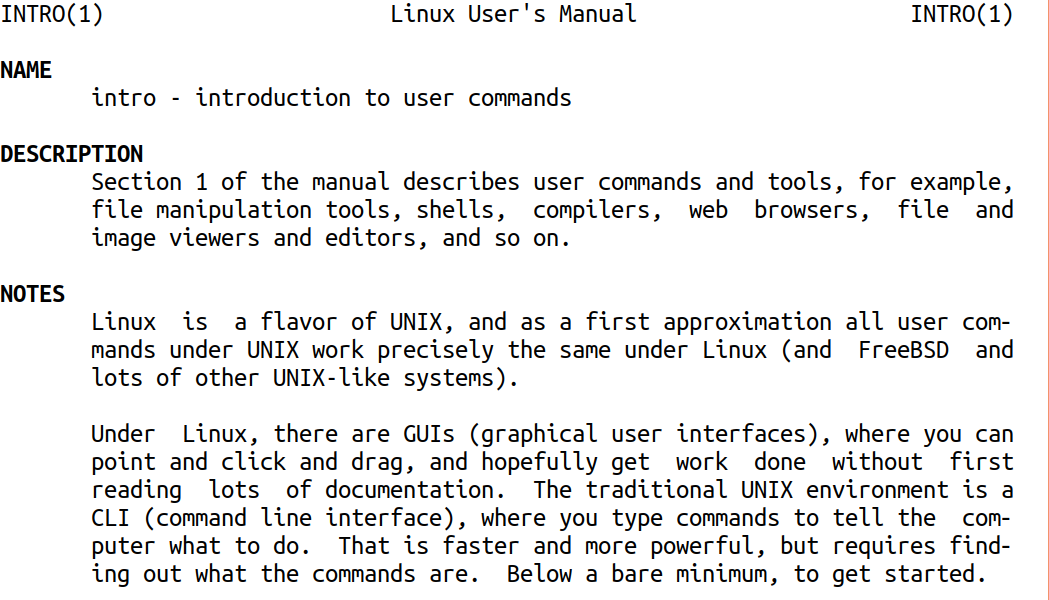


Рисунок 1.4 – Результат выполнения команды man -a intro

# 2. Последовательность действий по созданию программ для ОС Linux в среде CLion

Чтобы создать проект нужно выполнить следующие команды:

1. File🡪New Project🡪"C++ Executable"🡪Create🡪Location: "Project Path/Project Name"
2. Затем нажать правой кнопкой мыши на название нашего проекта, чтобы создать новый файл для программы:

Project 🡪 New 🡪 “C/C++ Source file” 🡪 “File name” 🡪 OK

1. Далее можно создавать программу

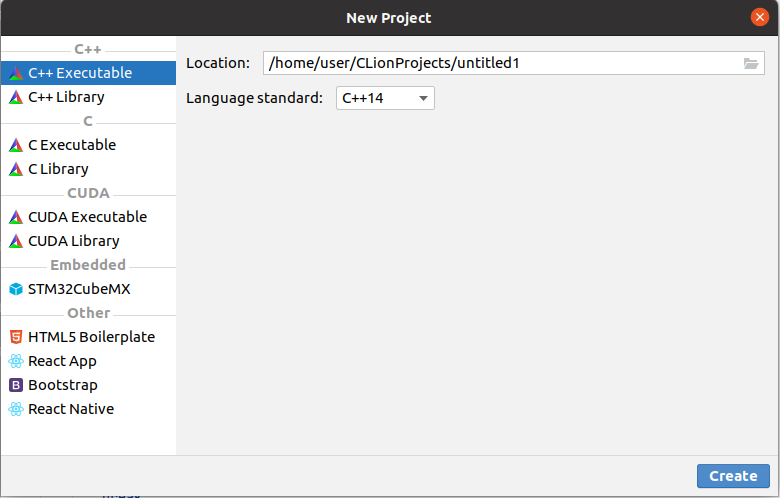


Рисунок 2.1 – Создание проекта в CLion

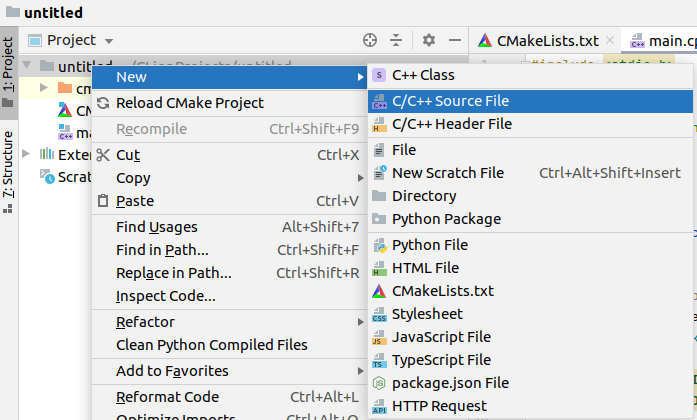


Рисунок 2.2 – Создание нового файла для программы в

СLion

3.Последовательность действий по отладке программ в среде разработки CLion

Для того чтобы выполнить отладку программы нужно:

1. Открыть вкладку Run 🡪 Debug 🡪 выбрать нужную конфигурацию, в которой находится программа.
2. Если обозначить точки останова в интересующих местах кода, то в процессе отладки можно двигаться по ним и отслеживать состояние данных.

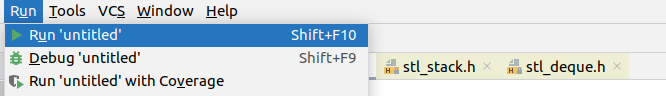


Рисунок 3.1 – Открытие вкладки Run для дальнейшей отладки программы

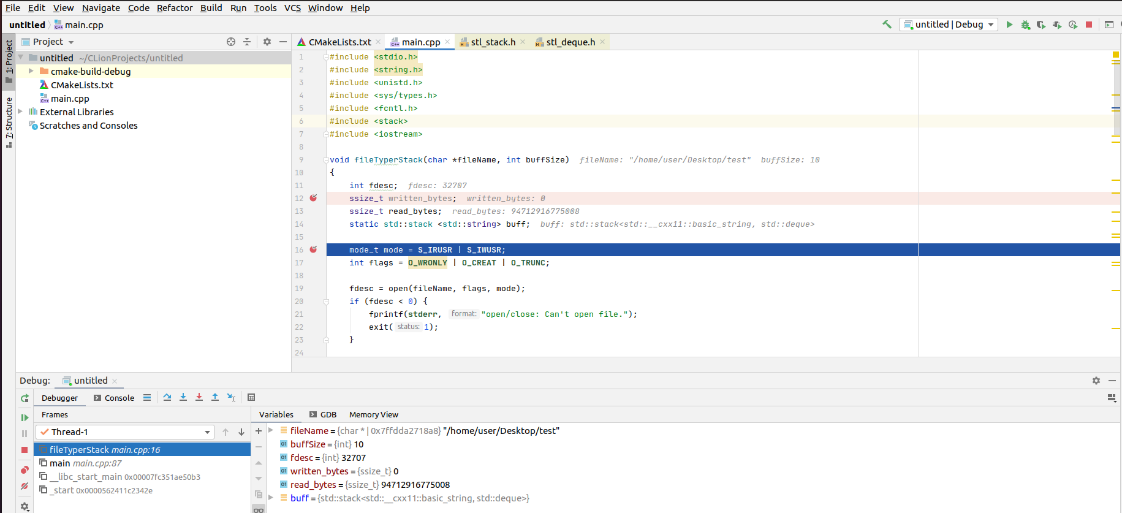


Рисунок 3.2 – Процесс отладки и отслеживание состояния данных во время прохода точки остановки

# 4.Приложение, реализующее безопасное чтение строк из стандартного потока ввода с их последующей записью в файл

Приложение, реализующее безопасное чтение строк из стандартного потока вывода с их последующей записью в файл представлено в листинге 1.

Листинг 1. Приложение, реализующее безопасное чтение строк и запись их в файл

void file\_typer(char \*file\_name, int buff\_size) {

int fdesc;

ssize\_t wr\_bytes;

ssize\_t rd\_bytes;

char str[buff\_size];

mode\_t mode = S\_IRUSR | S\_IWUSR;

int flags = O\_WRONLY | O\_CREAT | O\_TRUNC;

fdesc = open(file\_name, flags, mode);

if (fdesc < 0) {

fprintf(stderr, "Can't open file!\n");

exit(1);

}

while (true) {

fgets(str, sizeof(str), stdin);

if ((str[0]=='0') && (strlen(str)<3))

break;

else {

rd\_bytes = strlen(str);

wr\_bytes = write(fdesc, str, rd\_bytes);

}

}

}

Подаваемые параметры для программ, реализующих безопасное чтение строк из стандартного потока вывода с их последующей записью в файл в прямом и обратном порядке представлены на рисунке 4.1

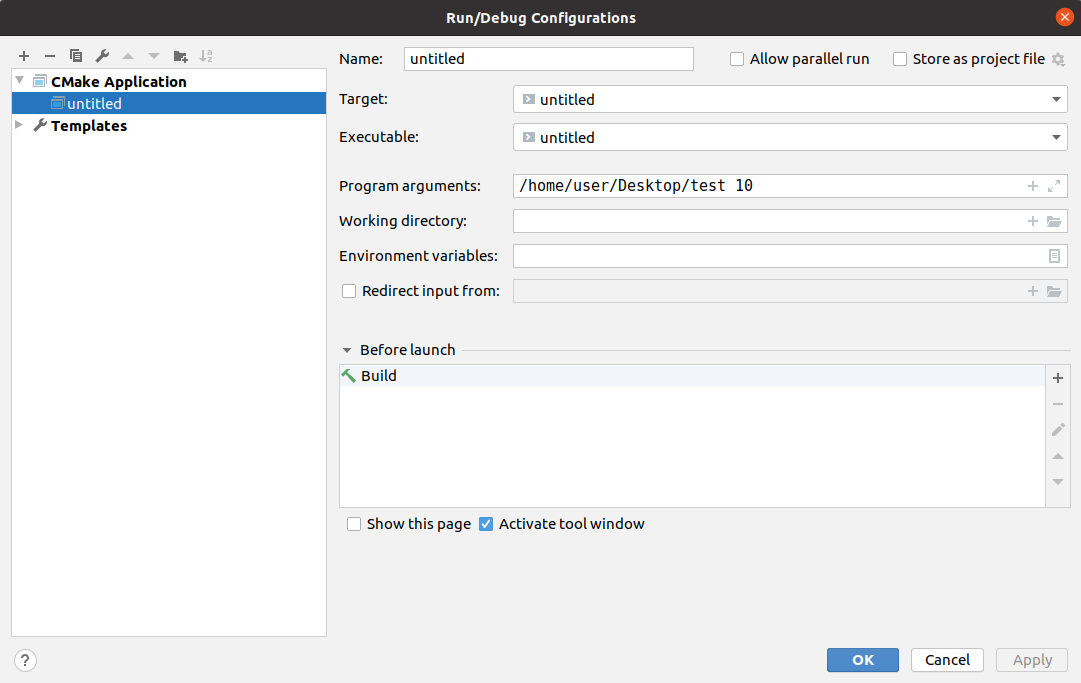


Рисунок 4.1 – Подаваемые параметры для задания 4 и 5

Результат выполнения приложения, реализующего безопасное чтение строк из стандартного потока вывода с их последующей записью в файл представлено на рисунке 4.2.

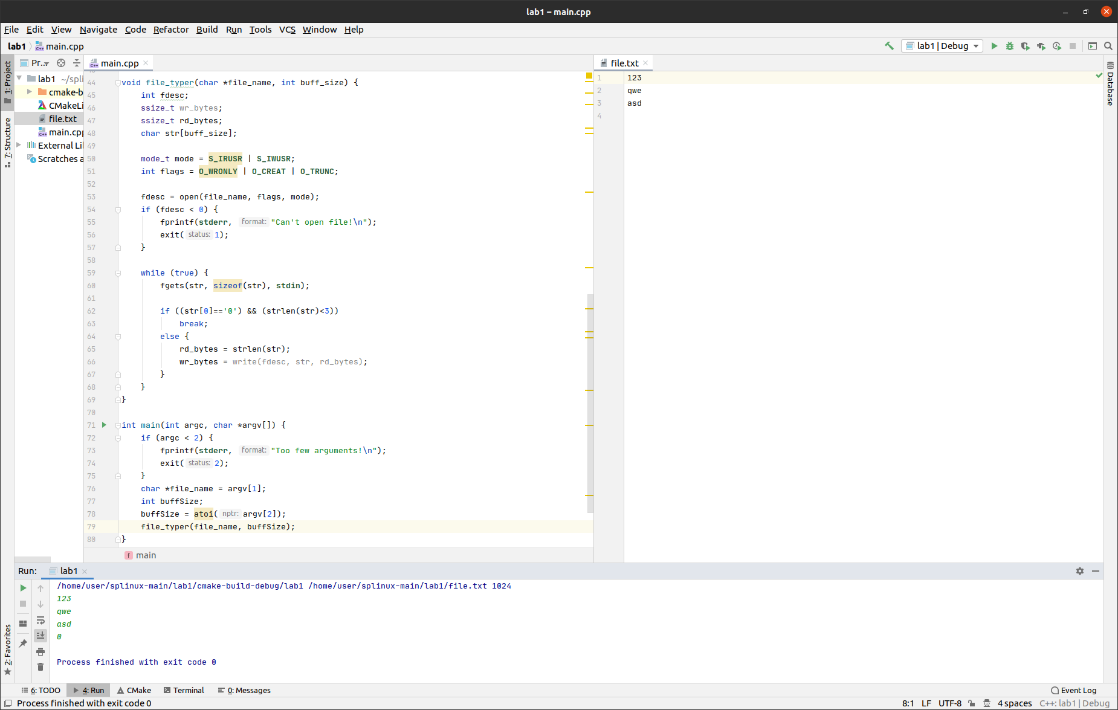


Рисунок 4.2 – Результат выполнения программы

# 5. Модифицированное приложение из пункта 4, записывающее строки в обратном порядке

Модифицированное приложение, реализующее безопасное чтение строк из стандартного потока вывода с их последующей записью в файл в обратном порядке представлено в листинге 2.

Листинг 2. Модифицированное приложение, реализующее безопасное чтение строк и запись их в файл в обратном порядке

void file\_typer\_stack(char \*fileName, int buff\_size) {

int fdesc;

ssize\_t wr\_bytes;

ssize\_t rd\_bytes;

std::stack <std::string> buff;

mode\_t mode = S\_IRUSR | S\_IWUSR;

int flags = O\_WRONLY | O\_CREAT | O\_TRUNC;

fdesc = open(fileName, flags, mode);

if (fdesc < 0) {

fprintf(stderr, "Can't open file!\n");

exit(1);

}

while (true) {

char str[buff\_size];

fgets(str, sizeof(str), stdin);

if ((str[0]=='0') && (strlen(str)<3))

break;

else {

buff.push(str);

}

}

while(!buff.empty()) {

std::string dest;

dest = buff.top();

rd\_bytes = dest.length();

wr\_bytes = write(fdesc, dest.c\_str(), rd\_bytes);

buff.pop();

}

}

Результат выполнения модифицированного приложения, реализующего безопасное чтение строк из стандартного потока вывода с их последующей записью в файл в обратном порядке представлено на рисунке 5.1.

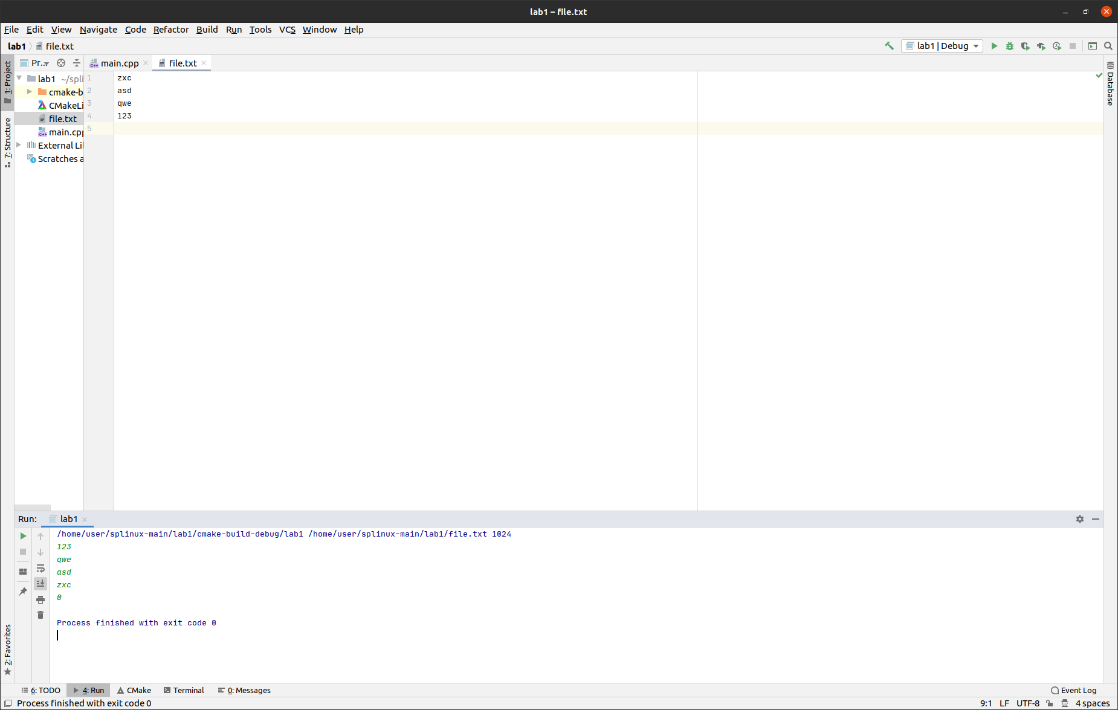


Рисунок 5.1 – Результат выполнения задания 5

# Вывод

1. Приобретены навыки по использованию системной справки man.
2. Проделаны основные этапы создания и отладки программ в среде разработки CLion.
3. Сформированы навыки по работе со строками из стандартного потока ввода на языке C.
4. Приобретены навыки чтения и записи потоковых файлов на языке C.