МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 8**

по дисциплине:” Системное программирование”

на тему: “Создание и использование библиотек в

Linux”

Выполнил**:** студент группы 10701321

Мацкевич М.Ю.

Принял**:** Кондратёнок Е.В.

Минск 2023

# Лабораторная работа № 8. Создание и использование библиотек в Linux

# Цель работы: Изучить и закрепить на практике создание и использование статически и динамически подключаемых библиотек в операционных системах семейства

# Linux

**Задание 1**

Для проверки остаточных знаний учеников после рождественских каникул,

учитель младших классов решил начинать каждый урок с того, чтобы задавать каждому ученику пример из таблицы умножения, но в классе, к примеру, 15 человек, а примеры среди них не должны повторяться. В помощь учителю разработайте универсальную библиотечную функцию, которая будет выводить на экран 15 (или N) случайных примеров из таблицы умножения (к примеру, от 2\*2 до 9\*9, потому что задания по умножению на 1 и на 10 – слишком просты). При этом среди 15 примеров не должно быть повторяющихся (примеры 2\*3 и 3\*2 и им подобные пары считать повторяющимися).

Алгоритм поиска данных комбинаций должен быть эффективным!!!

**Решение**

Напишем программу на с++

program.cpp

#include <iostream>

#include <vector>

#include <ctime>

#include <cstdlib>

#include <algorithm>

using namespace std;

vector<pair<int, int>> generateUniqueMultiplicationExamples(int n) {

vector<pair<int, int>> examples;

// Инициализация генератора случайных чисел

srand(static\_cast<unsigned>(time(nullptr)));

while (examples.size() < n) {

int num1 = rand() % 8 + 2; // Генерируем первый множитель от 2 до 9

int num2 = rand() % 8 + 2; // Генерируем второй множитель от 2 до 9

pair<int, int> example(num1, num2);

// Проверяем, что такой пример ещё не использовался

if (find(examples.begin(), examples.end(), example) == examples.end()) {

examples.push\_back(example);

}

}

return examples;

}

main.cpp

#include <iostream>

#include <vector>

#include <ctime>

#include <cstdlib>

#include <algorithm>

#include <dlfcn.h>

using namespace std;

int main() {

void\* lib = dlopen("libprogram.so", RTLD\_LAZY);

if (!lib) {

cerr << "Ошибка загрузки библиотеки: " << dlerror() << endl;

return 1;

}

typedef vector<pair<int, int>> (\*GenerateUniqueMultiplicationExamples)(int);

GenerateUniqueMultiplicationExamples generateUniqueMultiplicationExamples = reinterpret\_cast<GenerateUniqueMultiplicationExamples>(dlsym(lib, "\_Z36generateUniqueMultiplicationExamplesi"));

if (!generateUniqueMultiplicationExamples) {

cerr << "Ошибка загрузки функции: " << dlerror() << endl;

dlclose(lib);

return 1;

}

int n = 15; // Количество уникальных примеров, которые нужно сгенерировать

vector<pair<int, int>> uniqueExamples = generateUniqueMultiplicationExamples(n);

// Выводим сгенерированные примеры на экран

for (const auto& example : uniqueExamples) {

cout << example.first << " \* " << example.second << " = " << (example.first \* example.second) << endl;

}

return 0;

}

Напишем makefile, который будет собирать нашу программу

CC = g++

CFLAGS = -Wall -std=c++11

# Исходные файлы

SRCS = program.cpp

# Имя статической библиотеки

LIB = libprogram.a

# Имя исполняемого файла

EXEC = guess\_number\_game

all: $(EXEC)

$(EXEC): main.cpp $(LIB)

$(CC) $(CFLAGS) -o $@ $< -L. -lprogram

$(LIB): $(SRCS)

$(CC) $(CFLAGS) -c $^

ar rcs $@ $(SRCS)

.PHONY: clean

clean:

rm -f $(EXEC) $(LIB)

Имеется исходная структура нашей программы (рис.1)

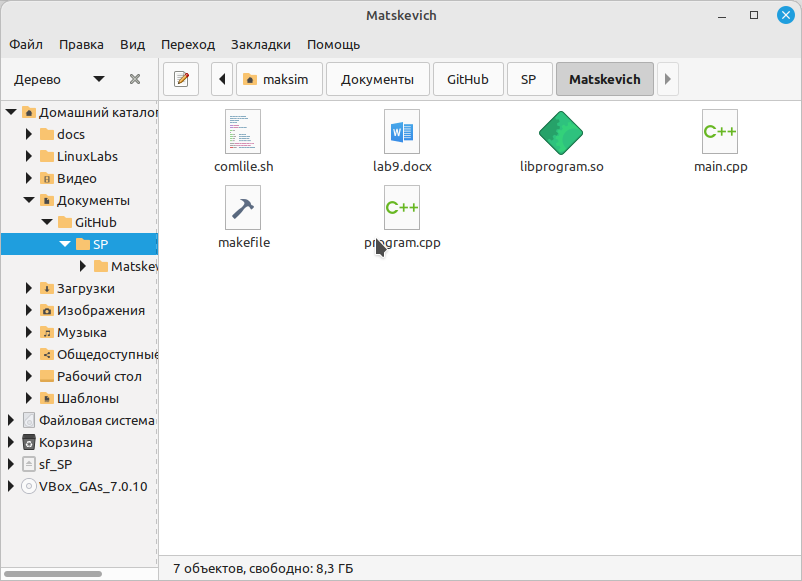


Рис.1 Исходная структура программы

Выполним команду make (рис.2) и получим новые файлы (рис.3)

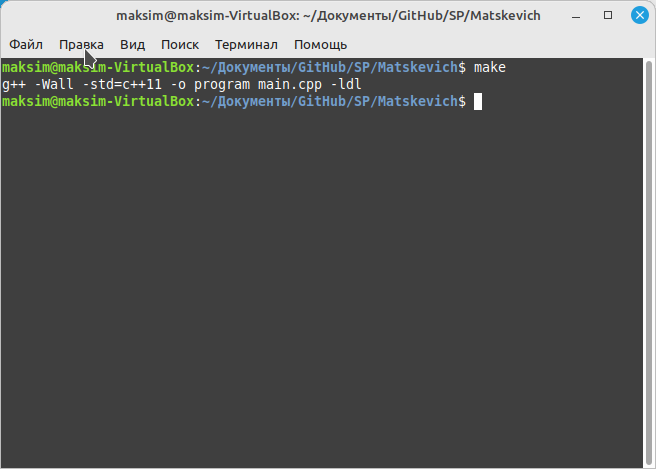


Рис.2 – Компиляция программы

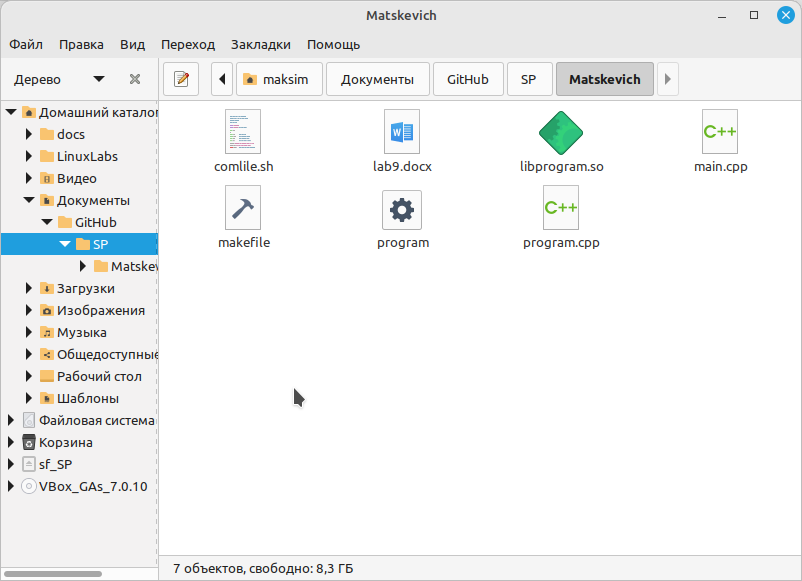


Рис.3 Новая структура программы

Выполним программу

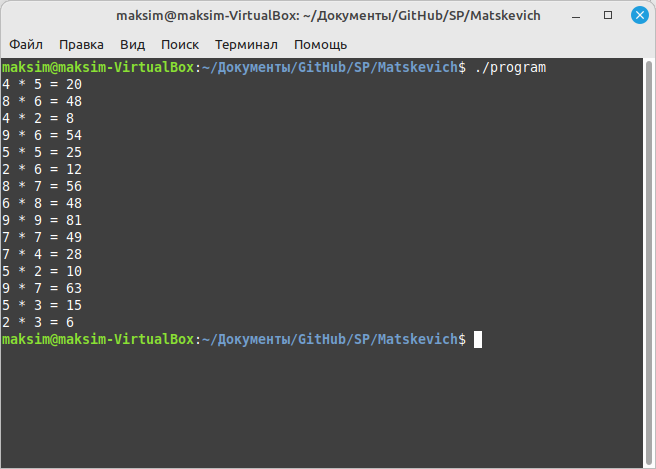


Рис.4 Результат выполнения программы

Вывод*:* В ходе выполнения лабораторной работы научились работать с библиотеками.

**Контрольные вопросы**

6. Чем отличается заголовочные файлы (файлы с расширением \*.h) от файлов

библиотек?

Заголовочные файлы \*.h содержат декларации функций, классов, переменных и других элементов программы, которые будут использоваться в других файлах программы или внешних библиотеках. Они обычно не содержат определений (реализаций) этих элементов, а только описывают, как они должны быть использованы.

Файлы библиотек (\*.lib, \*.dll, \*.a и т. д.) содержат уже скомпилированный объектный код и реализации функций, классов и других элементов программы, которые определены в заголовочных файлах. Они представляют собой готовые к использованию модули, которые можно подключить к программе для выполнения нужных функций.