Балтийский государственный технический университет «ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5
«Информационные системы и программная инженерия»

Лабораторная работа № 3
по дисциплине «Компьютерный практикум»
на тему **«Создание динамических библиотек при помощи набора компиляторов и утилит GCC и их применение.»**

Выполнил:
Студент Дубровский В.И
Группа И582

Преподаватель:
Вальштейн К. В.

Санкт-Петербург
2019

Цель работы:

Изучить процесс создания динамических библиотек при помощи набора компиляторов и утилит GCC и особенности их применения.

Задание:

Подсчитать число элементов матрицы Q (5x7) и массива R(57), кратных трем.

Путь созданных файлов:

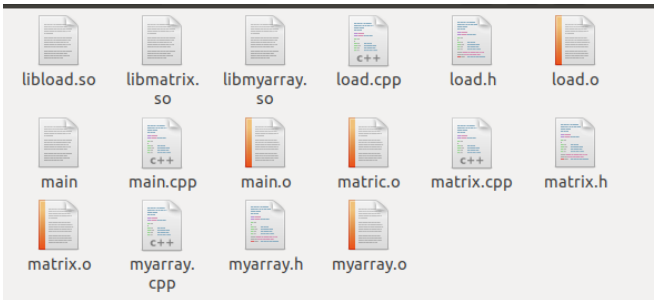

Рисунок 3.1 - Демонстрация расположения файлов

Команды использованные для компиляции динамической библиотеки:

Создаем динамическую библиотеку из myarray.cpp:

```
g++ -fPIC -c -o myarray.o myarray.cpp
g++ -shared -o libmyarray.so myarray.o
```

Создаем динамическую библиотеку из matrix.cpp:

```
g++ -fPIC -c -o matrix.o matrix.cpp
п++ -shared -o libmatrix.so com\3lab\matrix.o
```

Создаем динамическую библиотеку из load.cpp:

```
g++ -fPIC -c -o load.o load.cpp
g++ -shared -o libload.so load.o
```

Скомпилируем и запустим полученную программу:

```
g++ -fPIC -c main.cpp -o main.o
g++ -o main main.o -L. –lload
./main
```

Результаты тестирования:

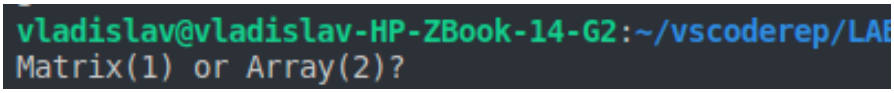

Рисунок 3.2 - Демонстрация меню выбора

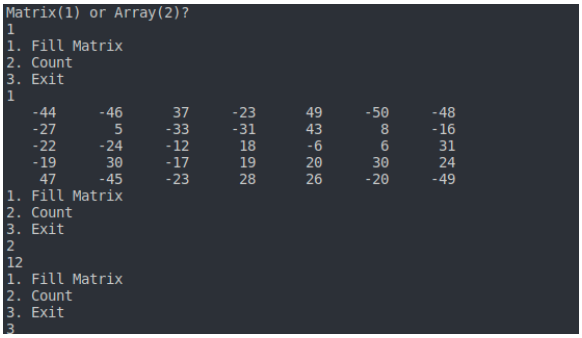

Рисунок 3.3 - Демонстрация работы с библиотекой массива

Текст программы:

main.cpp

```
#include <iostream>

#include "load.h"

constexpr auto N = 57;

constexpr auto N1 = 5;
constexpr auto N2 = 7;

constexpr auto X = -50;
constexpr auto Y = 50;

constexpr auto NUM = 3;

#ifdef WIN32
#define LIBMATRIX "matrix.dll"
#define LIBARRAY "myarray.dll"
#else
#define LIBMATRIX "libmatrix.so"
#define LIBARRAY "libmyarray.so"
#endif

int main()
{
    std::cout << "Matrix(1) or Array(2)?" << std::endl;

    int answer = 0;
    void* library = nullptr;
    std::cin >> answer;
    if(answer == 1)
    {
        int** matrix;
        matrix = new int*[N1];
        for(int i = 0; i < N1; i++)
        {
```

```
            matrix[i] = new int[N2];
        }


        library = loadDLL(LIBMATRIX);

        while(answer != 3)
        {
            std::cout << "1. Fill Matrix\n" <<
            "2. Count\n"  <<
            "3. Exit" << std::endl;
            std::cin >> answer;
            if(answer == 1)
            {
                auto             function             =             loadFunction<void
(*)(int**,int,int,int,int)>(library, "fillRandomMatrix");
                if(function != nullptr)
                {
                    function(matrix, N1, N2,X,Y);
                }
            }
            else if(answer == 2)
            {
                auto  function  =  loadFunction<int  (*)(int**,int,int,int)>(library,
"countMatrix");
                if(function != nullptr)
                {
                    std::cout << function(matrix, N1, N2, NUM) << std::endl;
                }

            }

        }

        for(int i=0; i<N1; i++)
        {
            delete[] matrix[i];
        }
        delete[] matrix;

    }
    else if(answer == 2)
    {
        int* array = new int[N];

        library = loadDLL(LIBARRAY);

        while(answer != 3)
        {
            std::cout << "1. Fill Array\n" <<
            "2. Count\n"  <<
            "3. Exit" << std::endl;
            std::cin >> answer;
            if(answer == 1)
            {
                auto  function  =  loadFunction<void  (*)(int  *,int,int,int)>(library,
"fillRandomArray");
                if(function != nullptr)
                {
                    function(array, N,X,Y);
                }
```

```
            }
            else if(answer == 2)
            {
                auto   function   =   loadFunction<int   (*)(int*,int,int)>(library,
"countArray");
                if(function != nullptr)
                {
                    std::cout << function(array, N, NUM) << std::endl;
                }

            }
        }

        delete[] array;
    }

    closeDLL(library);


    return 0;
}
```

load.cpp

```
#include "load.h"


#ifdef WIN32

void* loadDLL(const char* libraryName)
{
    void* library = LoadLibrary(libraryName);
    if(library == nullptr)
    {
        std::cerr << "Can't open library, name: " << libraryName << std::endl;
        return nullptr;
    }
    return library;
}

void closeDLL(void *library)
{
    FreeLibrary((HINSTANCE)library);
}
#else
void* loadDLL(const char* libraryName)
{
    void* library = dlopen(libraryName, RTLD_LAZY);
    if(library == nullptr)
    {
        std::cerr << "Can't open library, name: " << libraryName << std::endl
            << dlerror() << std::endl;
        return nullptr;
    }
    return library;
}

void closeDLL(void *library)
{
    dlclose(library);
}
```

```
#endif
```

matrix.cpp

```
#include "matrix.h"
#include <random>
#include <iostream>

void fillRandomMatrix(int** matrix, int N1, int N2,int x, int y)
{
    std::mt19937 mt_seed(time(NULL));
    std::uniform_int_distribution<int> uid(x,y);

    for(int i=0; i<N1; i++)
    {
        for(int j=0; j<N2; j++)
        {
            matrix[i][j] = uid(mt_seed);
            printf("%6i\t", matrix[i][j]);
        }
        std::cout << std::endl;
    }
}

int countMatrix(int** matrix, int N1, int N2, int num)
{
    int counter = 0;
    for(int i=0; i<N1; i++)
    {
        for(int j=0; j<N2; j++)
        {
            if(matrix[i][j]%num == 0)
            {
                counter++;
            }
        }
    }
    return counter;
}
```

array.cpp

```
#include "myarray.h"
#include <random>
#include <iostream>

void fillRandomArray(int* array, int N,int x, int y)
{
    std::mt19937 mt_seed(time(NULL));
    std::uniform_int_distribution<int> uid(x,y);

    for(int i=0; i<N; i++)
    {
        array[i] = uid(mt_seed);
        std::cout << array[i] << "  ";
    }
    std::cout << std::endl;
}
```

```
int countArray(int* array, int N, int num)
{
    int counter = 0;
    for(int i=0; i<N; i++)
    {
        if(array[i] % num == 0)
        {
            counter++;
        }
    }
    return counter;
}
```