

Лабораторная работа №5

Программирование. Язык СИ.

Многофайловый проект, условная компиляция,
утилита Make.

Задача:

Комплект 1: Многофайловый проект, условная компиляция, утилита Make.

- 1.1: Напишите программу из нескольких файлов (модулей), включая файл основной программы. Файлы должны содержать вынесенные отдельно функции для выделения памяти под динамические двумерные и одномерные массивы и функции для перемножения матриц. Собрать проект используя утилиту Make.

Список идентификаторов:

Имя	Смысл	Тип
row1	Количество строк в первой матрице	int
column1	Количество столбцов в первой матрице	int
row2	Количество строк во второй матрице	int
column2	Количество столбцов во второй матрице	int
mass1	Двумерный динамический массив (матрица 1)	double
mass2	Двумерный динамический массив (матрица 2)	double
result	Двумерный динамический массив (результат перемножения матриц 1 и 2)	double
matrix	Функция для выделения памяти под двумерный динамический массив	double
matrixFr	Функция для освобождения памяти	void
matrixDeploy	Функция для ручного заполнения массива	void
matrixApp	Функция для вывода двумерного массива	void
multiply	Функция для вычисления произведения матриц	double

Код программы (многофайловый проект):

Файл main.c:

```
#include <stdio.h>
#include "matrixFunc.h"
#include "matrixMult.h"

int main()
{
    int row1, column1, row2, column2;
    printf("Enter number of rows of matrix 1:");
    scanf("%d", &row1);
    printf("\nEnter number of columns of matrix 1:");
```

```

scanf("%d", &column1);

double **mass1 = matrix(row1, column1);
matrixDeploy(mass1, row1, column1);

printf("\n");

printf("Enter number of rows of matrix 2:");
scanf("%d", &row2);
printf("\nEnter number of columns of matrix 2:");
scanf("%d", &column2);

double **mass2 = matrix(row2, column2);
matrixDeploy(mass2, row2, column2);

printf("\nmatrix 1:\n");
matrixApp(mass1, row1, column1);
printf("\nmatrix 2:\n");
matrixApp(mass2, row2, column2);

double **result = multiply(mass1, row1, column1, mass2, row2, column2);
printf("\nThe result of multiplication: \n");
matrixApp(result, row1, column2);

printf("\nFreeing up memory for the matrix 1: ");
matrixFr(mass1, row1);
printf("\nFreeing up memory for the matrix 2: ");
matrixFr(mass2, row2);
printf("\nFreeing up memory for the resulting matrix: ");
matrixFr(result, row1);

return 0;
}

```

Файл matrixFunc.c:

```

#include <stdio.h>
#include <stdlib.h>
#include "matrixFunc.h"

double ** matrix(int rows, int columns)
{
    double **mass = (double **) malloc(rows * sizeof(double *));
    for (int i = 0; i < rows; i++)
    {
        mass[i] = (double *) malloc(columns * sizeof(double));
    }
    return(mass);
}

void matrixFr(double **mass, int rows)
{
    for (int i = 0; i < rows; i++)
    {
        free(mass[i]);
    }
    free(mass);
    printf("Success");
}

void matrixDeploy(double **mass, int rows, int columns)
{

```

```

        for (int i = 0; i < rows; i++)
        {
            double a;
            for (int j = 0; j < columns; j++)
            {
                printf("\nEnter element mass[%d][%d]=", i+1, j+1);
                scanf("%lf",&a);
                mass[i][j]=a;
            }
        }
    }

void matrixApp(double **mass, int rows, int columns)
{
    printf("\n");
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < columns; j++)
        {
            printf("%.11f ", mass[i][j]);
        }
        printf("\n");
    }
}

```

Файл matrixFunc.h:

```

#ifndef LABA5_MATRIXFUNC_H
#define LABA5_MATRIXFUNC_H

double ** matrix(int columns,int rows);
void matrixFr(double **mass, int rows);
void matrixDeploy(double **mass, int rows, int columns);
void matrixApp(double **mass, int rows, int columns);

#endif

```

Файл matrixMult.c:

```

#include <stdio.h>
#include "matrixFunc.h"
#include "matrixMult.h"

double **multiply(double **mass1, int row1, int column1, double **mass2, int row2, int column2)
{
    double **result = NULL;
    if (column1 == row2)
    {
        result = matrix(row1, column2);
        for (int i = 0; i < row1; i++)
        {
            for (int j = 0; j < column2; j++)
            {
                result[i][j] = 0;
                for (int p = 0; p < column1; p++)
                {
                    result[i][j] = result[i][j] + mass1[i][p] * mass2[p][j];
                }
            }
        }
    }
}

```

```

    }
}
return result;
}

```

Файл matrixMult.h:

```

#ifndef LABA5_MATRIXMULT_H
#define LABA5_MATRIXMULT_H

double **multiply(double **mass1, int row1, int column1, double **mass2, int
row2, int column2);

#endif

```

Результат выполнения программы:

```

Enter number of rows of matrix 1: 4
Enter number of columns of matrix 1: 3
Enter element mass[1][1]=1
Enter element mass[1][2]=2
Enter element mass[1][3]=3
Enter element mass[2][1]=4
Enter element mass[2][2]=5
Enter element mass[2][3]=6
Enter element mass[3][1]=7
Enter element mass[3][2]=8
Enter element mass[3][3]=9
Enter element mass[4][1]=10
Enter element mass[4][2]=11
Enter element mass[4][3]=12

Enter number of rows of matrix 2: 2
Enter number of columns of matrix 2: 5
Enter element mass[1][1]=1
Enter element mass[1][2]=2
Enter element mass[1][3]=3
Enter element mass[1][4]=4
Enter element mass[1][5]=5
Enter element mass[2][1]=6
Enter element mass[2][2]=7

```

```
Enter element mass[2][3]=2
```

```
Enter element mass[2][4]=3
```

```
Enter element mass[2][5]=4
```

```
Enter element mass[3][1]=7
```

```
Enter element mass[3][2]=8
```

```
Enter element mass[3][3]=2
```

```
Enter element mass[3][4]=5
```

```
Enter element mass[3][5]=6
```

```
matrix 1:
```

```
1.0 5.0 8.0
```

```
7.0 3.0 4.0
```

```
6.0 9.0 2.0
```

```
1.0 4.0 6.0
```

```
matrix 2:
```

```
1.0 8.0 3.0 4.0 9.0
```

```
6.0 5.0 2.0 3.0 4.0
```

```
7.0 8.0 2.0 5.0 6.0
```

```
The result of multiplication:
```

```
87.0 97.0 29.0 59.0 77.0
```

```
53.0 103.0 35.0 57.0 99.0
```

```
74.0 109.0 40.0 61.0 102.0
```

```
67.0 76.0 23.0 46.0 61.0
```

```
Freeing up memory for the matrix 1: Success
```

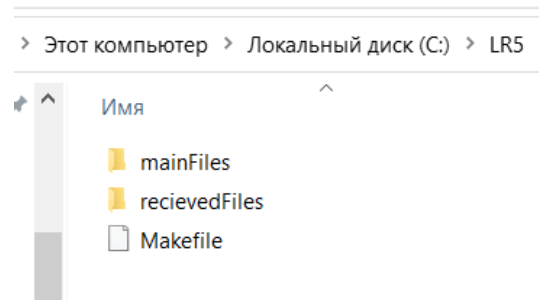
```
Freeing up memory for the matrix 2: Success
```

```
Freeing up memory for the resulting matrix: Success
```

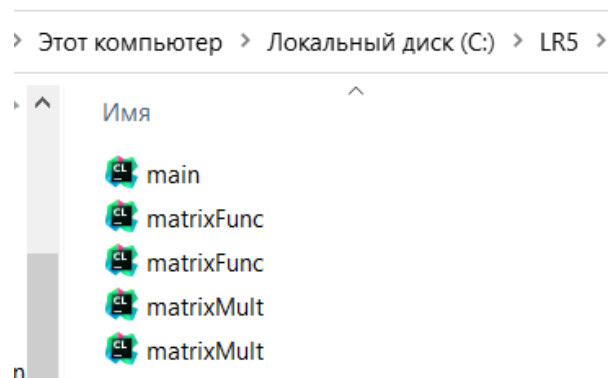
```
Process finished with exit code 0
```

Makefile для сборки проекта

Отдельная папка, в которой находится сам файл и две папки, одна из которых содержит исходные файлы с расширением “.c”.



mainFiles – папка, в которой лежат файлы проекта.



recievedFiles – папка, в которую будут помещены объектные файлы с расширением “.o”.

Рассмотрим сам Makefile

```
C:\LR5\Makefile - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксисы Опции Инструменты Макросы Запуск Плагины Вкладки ?
[Icons]
Makefile
1 Del = del
2
3 ProgramLR5 : main.o matrixFunc.o matrixMult.o
4     gcc recievedFiles/main.o recievedFiles/matrixFunc.o recievedFiles/matrixMult.o -o ProgramLR5
5
6 main.o : mainFiles/main.c
7     gcc -c mainFiles/main.c -o recievedFiles/main.o
8
9 matrixFunc.o : mainFiles/matrixFunc.c
10    gcc -c mainFiles/matrixFunc.c -o recievedFiles/matrixFunc.o
11
12 matrixMult.o : mainFiles/matrixMult.c
13    gcc -c mainFiles/matrixMult.c -o recievedFiles/matrixMult.o
14
15 clean:
16     $(DEL) recievedFiles\*.o ProgramLR5.exe
```

Первое правило:

```
ProgramLR5 : main.o matrixFunc.o matrixMult.o
```

```
gcc recievedFiles/main.o recievedFiles/matrixFunc.o recievedFiles/matrixMult.o -o ProgramLR5
```

Данное правило необходимо для сборки объектных файлов, в единый файл ProgramLR5, который в дальнейшем и будет запускаться через командную строку.

Через “ : ”, указаны файлы, которые необходимы для правильной компиляции нашего проекта (программы ProgramLR5).

Во второй строке, после указания компиляции через GCC, указаны пути к этим файлам из директории, в которой лежит Makefile (файлы будут лежать в папке recievedFiles). Флаг -o необходим для размещения результат компиляции в файл за ним, а именно в ProgramLR5.

Следующие 3 файла, необходимы для компиляции трёх объектных файлов, которые необходимы для сборки ProgramLR5 в первом правиле.

Второе правило:

```
main.o : mainFiles/main.c
```

```
gcc -c mainFiles/main.c -o recievedFiles/main.o
```

После “ : ”, указан файл, который будет компилироваться в объектный файл, и путь к нему из начальной директории.

Вторая строчка получает файл с расширением .c и с помощью gcc компилирует в файл с расширением .o. Берётся файл main.c из папки mainFiles (-c mainFiles/main.c) и создаётся объектный файл main.o, который помещается в папку recievedFiles, к которой обращается первое правило.

Третье правило:

```
matrixFunc.o : mainFiles/matrixFunc.c
```

```
gcc -c mainFiles/matrixFunc.c -o recievedFiles/matrixFunc.o
```

Аналогично со вторым правилом, данное правило необходимо для создания объектного файла matrixFunc.o.

Четвёртое правило:

matrixMult.o : mainFiles/matrixMult.c

gcc -c mainFiles/matrixMult.c -o recievedFiles/matrixMult.o

Создаётся файл matrixMult.o.

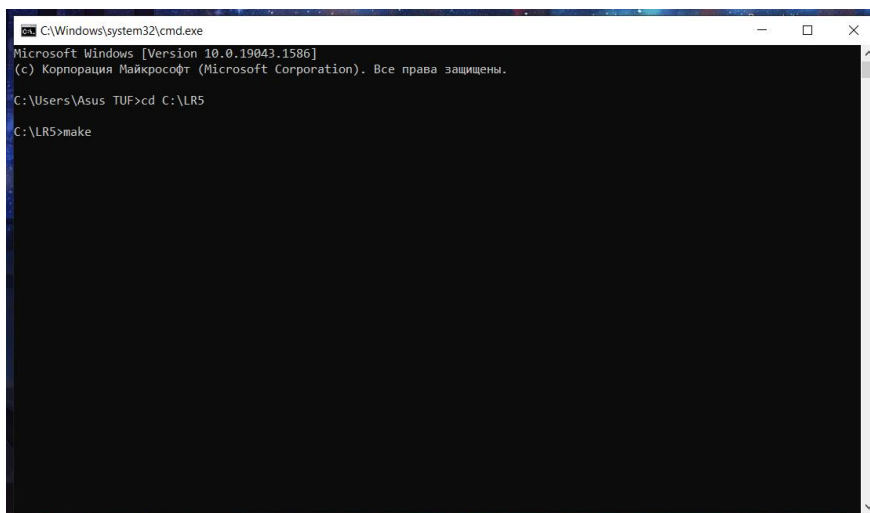
Пятое правило:

clean:

\$(DEL) recievedFiles*.o ProgramLR5.exe

Производит очистку указанной директории (recieveFiles) от файлов с расширением .o и удаляет файл ProgramLR5.exe. Необходимо для возможности компилировать проект снова, в случае изменения основных файлов.

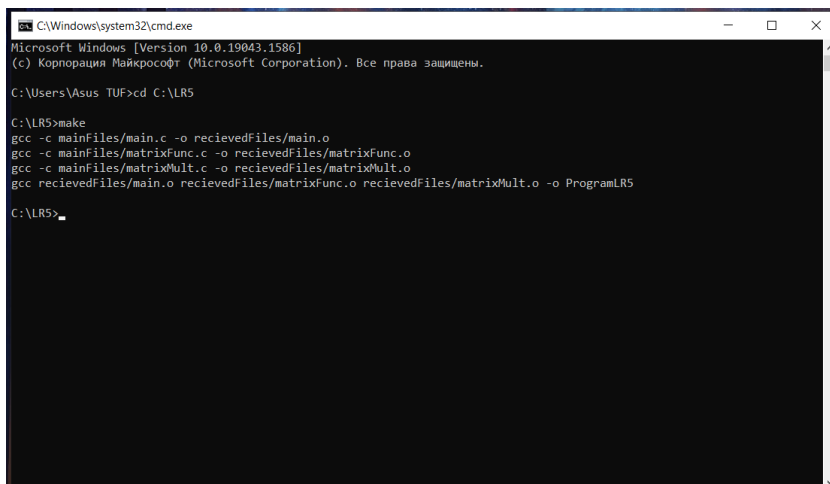
Компиляция через make:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1586]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Asus TUF>cd C:\LR5
C:\LR5>make
```

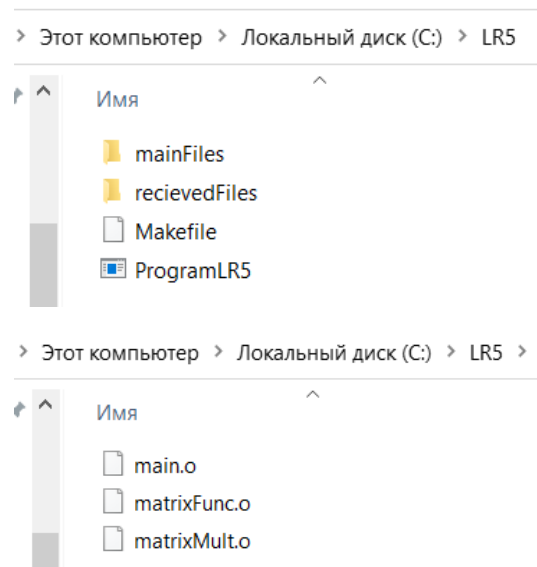
Заходим в директорию, в котором лежит сам файл Makefile и прописываем команду make.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1586]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Asus TUF>cd C:\LR5
C:\LR5>make
gcc -c mainFiles/main.c -o recievedFiles/main.o
gcc -c mainFiles/matrixFunc.c -o recievedFiles/matrixFunc.o
gcc -c mainFiles/matrixMult.c -o recievedFiles/matrixMult.o
gcc recievedFiles/main.o recievedFiles/matrixFunc.o recievedFiles/matrixMult.o -o ProgramLR5
C:\LR5>
```


Создаются 3 объектных файла, которые помещаются в папку receivedFiles и в основной директории создаётся скомпилированный файл ProgramLR5.



Вводим в консоли имя полученного основного файла. Программа запускается.

```
C:\Windows\system32\cmd.exe - ProgramLR5
Microsoft Windows [Version 10.0.19043.1586]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Asus TUF>cd C:\LR5

C:\LR5>make
gcc -c mainFiles/main.c -o recievedFiles/main.o
gcc -c mainFiles/matrixFunc.c -o recievedFiles/matrixFunc.o
gcc -c mainFiles/matrixMult.c -o recievedFiles/matrixMult.o
gcc recievedFiles/main.o recievedFiles/matrixFunc.o recievedFiles/matrixMult.o -o ProgramLR5

C:\LR5>ProgramLR5
Enter number of rows of matrix 1: _
```

Результат выполнения:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1586]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Asus TUF>cd C:\LR5

C:\LR5>make
gcc -c mainFiles/main.c -o recievedFiles/main.o
gcc -c mainFiles/matrixFunc.c -o recievedFiles/matrixFunc.o
gcc -c mainFiles/matrixMult.c -o recievedFiles/matrixMult.o
gcc recievedFiles/main.o recievedFiles/matrixFunc.o recievedFiles/matrixMult.o -o ProgramLR5

C:\LR5>ProgramLR5
Enter number of rows of matrix 1:3

Enter number of columns of matrix 1:1

Enter element mass[1][1]=4

Enter element mass[2][1]=5

Enter element mass[3][1]=6

Enter number of rows of matrix 2:1

Enter number of columns of matrix 2:2

Enter element mass[1][1]=7

Enter element mass[1][2]=4

matrix 1:

4.0
5.0
6.0

matrix 2:

7.0 4.0

The result of multiplication:

28.0 16.0
35.0 20.0
42.0 24.0

Freeing up memory for the matrix 1: Success
Freeing up memory for the matrix 2: Success
Freeing up memory for the resulting matrix: Success
C:\LR5>
```