1. Тема лабораторной работы: Препроцессор, модульная сборка программы

2. Постановка задачи:

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.

4. Список идентификаторов

Имя	Тип	Смысл
n1	int	Количество строк первой матрицы
m1	int	Количество столбцов первой матрицы
n2	int	Количество строк второй матрицы
m2	int	Количество столбцов второй матрицы
n	int	Размер массива
A	int**	Матрица A
B	int**	Матрица B
C	int**	Результат умножения матрицы A на матрицу B (Матрица C)
a	int*	Массив a
i	int	Параметр цикла
j	int	Параметр цикла

5. Код программы

```
#include <stdio.h>
#include <stdlib.h>
#include "matrix.h"
#include "array.h"

int main(void) {
    int n1, m1, n2, m2, n;
```

```
    printf("n of 1st matrix = ");
    scanf("%d", &n1);
    printf("m of 1st matrix = ");
    scanf("%d", &m1);
    printf("Fill A-matrix:\n");
    int **A = createAndFillMatrix(n1, m1);

    printf("n of 2nd matrix = ");
    scanf("%d", &n2);
    printf("m of 2nd matrix = ");
    scanf("%d", &m2);
    printf("Fill B-matrix:\n");
    int **B = createAndFillMatrix(n2, m2);

    printf("A * B = C-matrix:\n");
      int **C = multiplicationMatrix(A, B, n1, m2,
n2);
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < m2; j++) {
            printf("[%d][%d] = %d", i, j, C[i][j]);
        printf("\n");
        }
    }

    freeMatrix(A, n1);
    freeMatrix(B, n2);
    freeMatrix(C, n1);

    printf("Enter size of array: ");
    scanf("%d", &n);
    int *a = createAndFillArray(n);

    printf("Sort increase:\n");
    sortArrayIncrease(a, n);
```

```
    printArray(a, n);

    free(a);
    return 0;
}
```

4. Список идентификаторов

Имя	Тип	Смысл
n1	int	Количество строк первой матрицы
m1	int	Количество столбцов первой матрицы
n2	int	Количество строк второй матрицы
m2	int	Количество столбцов второй матрицы
n	int	Размер массива
A	int**	Матрица A
B	int**	Матрица B
C	int**	Результат умножения матрицы A на матрицу B (Матрица C)
a	int*	Массив a
i	int	Параметр цикла
j	int	Параметр цикла

5. Код программы *main.c*

```
#include <stdio.h>
#include <stdlib.h>
#include "matrix.h"
#include "array.h"

int main(void) {
    int n1, m1, n2, m2, n;

    printf("n of 1st matrix = ");
```

```
    scanf("%d", &n1);
    printf("m of 1st matrix = ");
    scanf("%d", &m1);
    printf("Fill A-matrix:\n");
    int **A = createAndFillMatrix(n1, m1);

    printf("n of 2nd matrix = ");
    scanf("%d", &n2);
    printf("m of 2nd matrix = ");
    scanf("%d", &m2);
    printf("Fill B-matrix:\n");
    int **B = createAndFillMatrix(n2, m2);

    printf("A * B = C-matrix:\n");
      int **C = multiplicationMatrix(A, B, n1, m2,
n2);
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < m2; j++) {
            printf("[%d][%d] = %d", i, j, C[i][j]);
        printf("\n");
        }
    }

    freeMatrix(A, n1);
    freeMatrix(B, n2);
    freeMatrix(C, n1);

    printf("Enter size of array: ");
    scanf("%d", &n);
    int *a = createAndFillArray(n);

    printf("Sort increase:\n");
    sortArrayIncrease(a, n);
    printArray(a, n);
```

```
    free(a);
    return 0;
}
```

4. Список идентификаторов

Имя	Тип	Смысл
createAndFillMatrix	int**	Функция выделения памяти под матрицу и её заполнение
freeMatrix	void	Функция освобождения памяти из-под матрицы
multiplicationMatrix	int**	Функция перемножения матриц
n	int	Аргумент функции (количество строк)
m	int	Аргумент функции (количество столбцов)
M	int**	Аргумент функции / возвращаемое значение из функции (матрица)
M1	int**	Аргумент функции (матрица 1)
M2	int**	Аргумент функции (матрица 2)
l	int	Аргумент функции (параметр цикла)
q	int	Аргумент функции (параметр цикла)
k	int	Аргумент функции (параметр цикла)
i	int	Параметр цикла
j	int	Параметр цикла
R	int**	Матрица, полученная в результате умножения / возвращается из функции multiplicationMatrix

5. Код программы *matrix.h*

```
#define matrix_h

int **createAndFillMatrix(int n, int m);
void freeMatrix(int**, int);
int** multiplicationMatrix(int** M1, int** M2, int
l, int q, int k);
```

5. Код программы *matrix.c*

```
#include <stdlib.h>
#include <stdio.h>
#include "matrix.h"
```

```
int **createAndFillMatrix(int n, int m) {
    int** M = (int**)malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++)
        M[i] = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            printf("[%d][%d] = ", i, j);
            scanf("%d", &M[i][j]);
        }
    return M;
}

void freeMatrix(int** M, int n) {
    for (int i = 0; i < n; i++)
        free(M[i]);
    free(M);
}

int** multiplicationMatrix(int** M1, int** M2, int
l, int q, int k) {
    int **R = (int**)malloc(l * sizeof(int*));
    for (int i = 0; i < l; i++)
        R[i] = (int*)malloc(q*sizeof(int));
    for (int i = 0; i < l; i++)
        for (int j = 0; j < q; j++) {
            R[i][j] = 0;
            for (int r = 0; r < k; r++)
                R[i][j] += M1[i][r] * M2[r][j];
        }
    return R;
}
```

4. Список идентификаторов

Имя	Тип	Смысл

createAndFillArray	int*	Функция выделения памяти под массив и его заполнение
sortArrayIncrease	void	Функция сортировки массива
printArray	void	Функция вывода массива на экран
arr	int*	Аргумент функций / возвращаемое значение из функции (массив)
n	int	Аргумент функции / размер массива
i	int	Параметр цикла
newElement	int	Вспомогательная переменная при сортировки
location	int	Вспомогательная переменная при сортировки

5. Код программы *array.h*

```
#define array_h

int* createAndFillArray(int n);
void sortArrayIncrease(int *arr, int n);
void sortArrayDecrease(int *arr, int n);
void printArray(int *arr, int n);
```

5. Код программы *array.c*

```
#include <stdio.h>
#include <stdlib.h>
#include "array.h"
 int* createAndFillArray(int n) {
    int *arr = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        printf("[%d] = ", i);
        scanf("%d", &arr[i]);
    }
    return arr;
}
void sortArrayIncrease(int *arr, int n) {
    for (int i = 1; i < n; i++) {
```

```
        int newElement = arr[i];
        int location = i - 1;
            while(location >= 0 && arr[location] >
newElement) {
            arr[location + 1] = arr[location];
            location = location - 1;
        }
        arr[location + 1] = newElement;
    }
}

void printArray(int *arr, int n) {
    for (int i = 0; i < n; i++)
        printf("a[%d] = %d\n", i, arr[i]);
}
```

6. Результаты выполненной работы

```
n of 1st matrix = 2
m of 1st matrix = 2
Fill A-matrix:
[0][0] = 1
[0][1] = 2
[1][0] = 3
[1][1] = 4
n of 2nd matrix = 2
m of 2nd matrix = 2
Fill B-matrix:
[0][0] = 1
[0][1] = 2
[1][0] = 3
[1][1] = 4
A * B = C-matrix:
[0][0] = 7
[0][1] = 10
[1][0] = 15
[1][1] = 22
Enter size of array: 5
[0] = 4
[1] = 1
[2] = 2
[3] = 5
[4] = 3
Sort increase:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```