­МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

(НГТУ)

Институт радиоэлектроники и информационных технологий

Кафедра «Информационные радиосистемы»

**Контрольная работа по дисциплине  
«Информационные технологии»**

Направление подготовки: \_\_\_\_11.03.01 Радиотехника\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*код и наименование направления подготовки*

Выполнил:

Студент гр. 24-Рз\_\_\_\_\_\_\_\_\_\_\_\_Иванов И.И.

*(группа) (подпись)*

Проверил:

доцент кафедры ИРС \_\_\_\_\_\_\_\_\_\_ Балашова Д.М.

*(подпись)*

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_

Дата: «\_\_\_» \_\_\_\_\_\_\_\_\_\_2024 г.

Нижний Новгород, 2024

**Задание 1**

**Вариант**

**Блок-схема алгоритма**

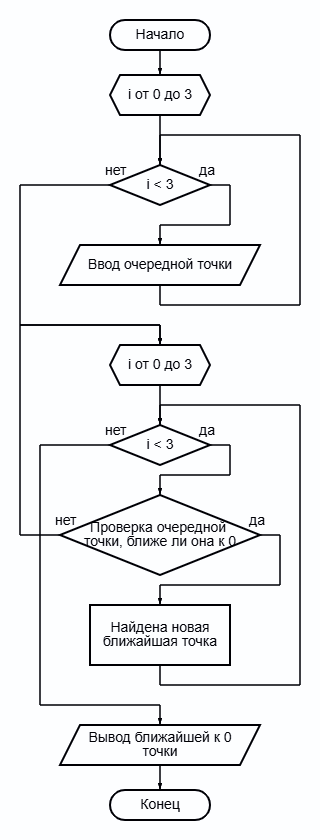
**Листинг программного кода**

**Задание 2. Часть 1**

**Вариант 11**

На оси ОХ расположены три точки х1 > 0, x2 > 0, x3 > 0. Определить, какая из данных точек расположена ближе к началу координат.

**Блок-схема алгоритма**

****

**Листинг программного кода**

#include <stdio.h>

#include <stdlib.h>

#define NUMBER\_OF\_POINTS 3

int main(){

    int x[NUMBER\_OF\_POINTS];

    int closest\_value = 99999;

    int closest\_index = 0;

    for (int i = 0; i < NUMBER\_OF\_POINTS; ++i){

        printf("Input a dot (integer) %d\n", i+1);

        scanf("%d", &x[i]);

    }

    for(int i = 0; i < NUMBER\_OF\_POINTS; ++i){

        if (abs(x[i]) < abs(closest\_value)){

            closest\_value = x[i];

            closest\_index = i+1;

        }

    }

    printf("Closest is dot %d, with value of %d\n", closest\_index, closest\_value);

    return 0;

}

**Задание 2. Часть 2**

**Вариант 12**

Сформируйте массив из значений полинома Лаггера при n = 7; x = 0.5.

**Блок-схема алгоритма**

****

**Листинг программного кода**

#include <stdio.h>

int main(){

    double x = 0.5;

    int n\_limit = 7;

    double L[n\_limit];

    L[0] = 1;

    printf("L1(%.1lf) = %lf\n", x, L[0]);

    L[1] = 1 - x;

    printf("L2(%.1lf) = %lf\n", x, L[1]);

    for (int n = 2; n < n\_limit; n++) {

        L[n] = (x - (2 \* n + 1)) \* L[n - 1] - (n - 1) \* (n - 1) \* L[n - 2];

        printf("L%d(%.1lf) = %lf\n", n+1, x, L[n]);

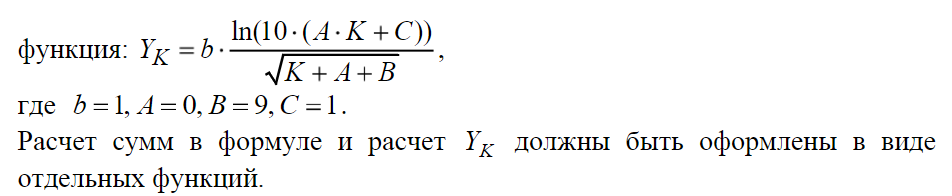
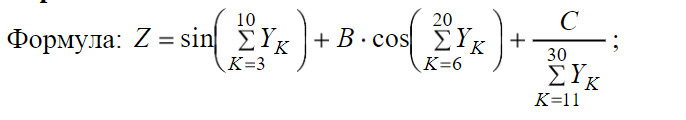
    }

    return 0;

}

**Задание 3**

**Вариант 4**

****

**Листинг программного кода**

#include <stdio.h>

#include <math.h>

double calculate\_Yk(int k, double A, double B, double C, double b) {

    return b \* (log(10.0 \* (A \* k + C)) / sqrt(k + A + B));

}

double calculate\_summation(int start, int end, double A, double B, double C, double b) {

    double sum = 0.0;

    for (int k = start; k <= end; k++) {

        sum += calculate\_Yk(k, A, B, C, b);

    }

    return sum;

}

int main() {

    double A = 0.0;

    double B = 9.0;

    double C = 1.0;

    double b = 1.0;

    double sum1 = calculate\_summation(3, 10, A, B, C, b);

    double sum2 = calculate\_summation(6, 20, A, B, C, b);

    double sum3 = calculate\_summation(11, 30, A, B, C, b);

    double Z = sin(sum1) + B \* cos(sum2) + (C / sum3);

    printf("Z = %lf\n", Z);

    return 0;

}

**Задание 4 - 1**

**Вариант 7**

Осуществить транспонирование квадратной матрицы размерностью 6х6. Матрицу инициализировать в программе.

**Листинг программного кода**

Header file:

#include <stdio.h>

#define SIZE 6

void matrix\_init(int matrix[SIZE][SIZE]);

void matrix\_transpose(int matrix[SIZE][SIZE], int transposed[SIZE][SIZE]);

void matrix\_print(int matrix[SIZE][SIZE]);

C file:

#include "4a.h"

void matrix\_init(int matrix[SIZE][SIZE]) {

    int value = 1;

    for (int i = 0; i < SIZE; i++) {

        for (int j = 0; j < SIZE; j++) {

            matrix[i][j] = value++;

        }

    }

}

void matrix\_transpose(int matrix[SIZE][SIZE], int transposed[SIZE][SIZE]) {

    for (int i = 0; i < SIZE; i++) {

        for (int j = 0; j < SIZE; j++) {

            transposed[j][i] = matrix[i][j];

        }

    }

}

void matrix\_print(int matrix[SIZE][SIZE]) {

    for (int i = 0; i < SIZE; i++) {

        for (int j = 0; j < SIZE; j++) {

            printf("%4d", matrix[i][j]);

        }

        printf("\n");

    }

}

int main() {

    int matrix\_original[SIZE][SIZE];

    int matrix\_transposed[SIZE][SIZE];

    matrix\_init(matrix\_original);

    printf("Original Matrix:\n");

    matrix\_print(matrix\_original);

    matrix\_transpose(matrix\_original, matrix\_transposed);

    printf("\nTransposed Matrix:\n");

    matrix\_print(matrix\_transposed);

    return 0;

}

**Задание 4 - 2**

**Вариант 7**

Осуществить транспонирование квадратной матрицы размерностью 6х6. Матрицу инициализировать в программе.

**Листинг программного кода**

Header file:

#include <stdio.h>

void matrix\_init(int \*matrix, int size);

void matrix\_transpose(int \*matrix, int \*transposed, int size);

void matrix\_print(int \*matrix, int size);

C file:

#include "4p.h"

#define SIZE 6

void matrix\_init(int \*matrix, int size) {

    int value = 1;

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            \*(matrix + i \* size + j) = value++;

        }

    }

}

void matrix\_transpose(int \*matrix, int \*transposed, int size) {

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            \*(transposed + j \* size + i) = \*(matrix + i \* size + j);

        }

    }

}

void matrix\_print(int \*matrix, int size) {

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            printf("%4d", \*(matrix + i \* size + j));

        }

        printf("\n");

    }

}

int main() {

    int matrix\_original[SIZE][SIZE];

    int matrix\_transposed[SIZE][SIZE];

    matrix\_init((int \*)matrix\_original, SIZE);

    printf("\nOriginal Matrix:\n");

    matrix\_print((int \*)matrix\_original, SIZE);

    matrix\_transpose((int \*)matrix\_original, (int \*)matrix\_transposed, SIZE);

    printf("\nTransposed Matrix:\n");

    matrix\_print((int \*)matrix\_transposed, SIZE);

    return 0;

}

**Задание 5 - 1**

**Вариант 8**

Таблица содержит геометрические точки, заданные в полярных координатах (α, R). Определить прямоугольник в плоскости Oxy, в который попадают все указанные точки x=Rcosα, y=Rsinα . Стороны прямоугольника выбрать ориентированными вдоль координатных осей.

**Листинг программного кода**

Header file:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX\_POINTS 100

typedef struct {

    double alpha;

    double r;

} PolarPoint;

typedef struct {

    double min\_x;

    double max\_x;

    double min\_y;

    double max\_y;

} Rectangle;

void polar\_to\_cartesian(PolarPoint polar, double cartesian[2]);

int input\_polar\_points(PolarPoint points[]);

Rectangle calculate\_bounding\_rectangle(PolarPoint points[], int num\_points);

void output\_rectangle(Rectangle rect);

C file:

#include "5a.h"

void polar\_to\_cartesian(PolarPoint polar, double cartesian[2]) {

    double alpha\_rad = polar.alpha \* M\_PI / 180.0;

    cartesian[0] = polar.r \* cos(alpha\_rad);

    cartesian[1] = polar.r \* sin(alpha\_rad);

}

int input\_polar\_points(PolarPoint points[]) {

    int num\_points;

    printf("Enter the number of points (up to %d): ", MAX\_POINTS);

    if (scanf("%d", &num\_points) != 1 || num\_points <= 0 || num\_points > MAX\_POINTS) {

        printf("Invalid number of points.\n");

        return -1;

    }

    printf("Enter the polar coordinates (alpha r) for each point:\n");

    for (int i = 0; i < num\_points; i++) {

        if (scanf("%lf %lf", &points[i].alpha, &points[i].r) != 2) {

            printf("Invalid input for point %d.\n", i + 1);

            return -1;

        }

    }

    return num\_points;

}

Rectangle calculate\_bounding\_rectangle(PolarPoint points[], int num\_points) {

    Rectangle rect;

    rect.min\_x = rect.min\_y = INFINITY;

    rect.max\_x = rect.max\_y = -INFINITY;

    double cartesian[2];

    for (int i = 0; i < num\_points; i++) {

        polar\_to\_cartesian(points[i], cartesian);

        if (cartesian[0] < rect.min\_x) rect.min\_x = cartesian[0];

        if (cartesian[0] > rect.max\_x) rect.max\_x = cartesian[0];

        if (cartesian[1] < rect.min\_y) rect.min\_y = cartesian[1];

        if (cartesian[1] > rect.max\_y) rect.max\_y = cartesian[1];

    }

    return rect;

}

void output\_rectangle(Rectangle rect) {

    printf("Bounding Rectangle:\n");

    printf("Min X: %.2lf\n", rect.min\_x);

    printf("Max X: %.2lf\n", rect.max\_x);

    printf("Min Y: %.2lf\n", rect.min\_y);

    printf("Max Y: %.2lf\n", rect.max\_y);

}

int main() {

    PolarPoint points[MAX\_POINTS];

    int num\_points = input\_polar\_points(points);

    if (num\_points == -1) {

        return 1;

    }

    Rectangle rect = calculate\_bounding\_rectangle(points, num\_points);

    output\_rectangle(rect);

    return 0;

}

**Задание 5 - 2**

**Вариант 8**

Таблица содержит геометрические точки, заданные в полярных координатах (α, R). Определить прямоугольник в плоскости Oxy, в который попадают все указанные точки x=Rcosα, y=Rsinα . Стороны прямоугольника выбрать ориентированными вдоль координатных осей.

**Листинг программного кода**

Header file:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX\_POINTS 100

typedef struct {

    double alpha;

    double r;

} PolarPoint;

typedef struct {

    double min\_x;

    double max\_x;

    double min\_y;

    double max\_y;

} Rectangle;

void polar\_to\_cartesian(PolarPoint \*polar, double \*x, double \*y);

int input\_polar\_points(PolarPoint \*points);

void calculate\_bounding\_rectangle(PolarPoint \*points, int num\_points, Rectangle \*rect);

void output\_rectangle(Rectangle \*rect);

C file:

#include "5p.h"

void polar\_to\_cartesian(PolarPoint \*polar, double \*x, double \*y) {

    double alpha\_rad = polar->alpha \* M\_PI / 180.0;

    \*x = polar->r \* cos(alpha\_rad);

    \*y = polar->r \* sin(alpha\_rad);

}

int input\_polar\_points(PolarPoint \*points) {

    int num\_points;

    printf("Enter the number of points (up to %d): ", MAX\_POINTS);

    if (scanf("%d", &num\_points) != 1 || num\_points <= 0 || num\_points > MAX\_POINTS) {

        printf("Invalid number of points.\n");

        return -1;

    }

    printf("Enter the polar coordinates (alpha r) for each point:\n");

    for (int i = 0; i < num\_points; i++) {

        if (scanf("%lf %lf", &points[i].alpha, &points[i].r) != 2) {

            printf("Invalid input for point %d.\n", i + 1);

            return -1;

        }

    }

    return num\_points;

}

void calculate\_bounding\_rectangle(PolarPoint \*points, int num\_points, Rectangle \*rect) {

    rect->min\_x = rect->min\_y = INFINITY;

    rect->max\_x = rect->max\_y = -INFINITY;

    double x, y;

    for (int i = 0; i < num\_points; i++) {

        polar\_to\_cartesian(&points[i], &x, &y);

        if (x < rect->min\_x) rect->min\_x = x;

        if (x > rect->max\_x) rect->max\_x = x;

        if (y < rect->min\_y) rect->min\_y = y;

        if (y > rect->max\_y) rect->max\_y = y;

    }

}

void output\_rectangle(Rectangle \*rect) {

    printf("Bounding Rectangle:\n");

    printf("Min X: %.2lf\n", rect->min\_x);

    printf("Max X: %.2lf\n", rect->max\_x);

    printf("Min Y: %.2lf\n", rect->min\_y);

    printf("Max Y: %.2lf\n", rect->max\_y);

}

int main() {

    PolarPoint points[MAX\_POINTS];

    Rectangle rect;

    int num\_points = input\_polar\_points(points);

    if (num\_points == -1) {

        return 1;

    }

    calculate\_bounding\_rectangle(&points, num\_points, &rect);

    output\_rectangle(&rect);

    return 0;

}