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

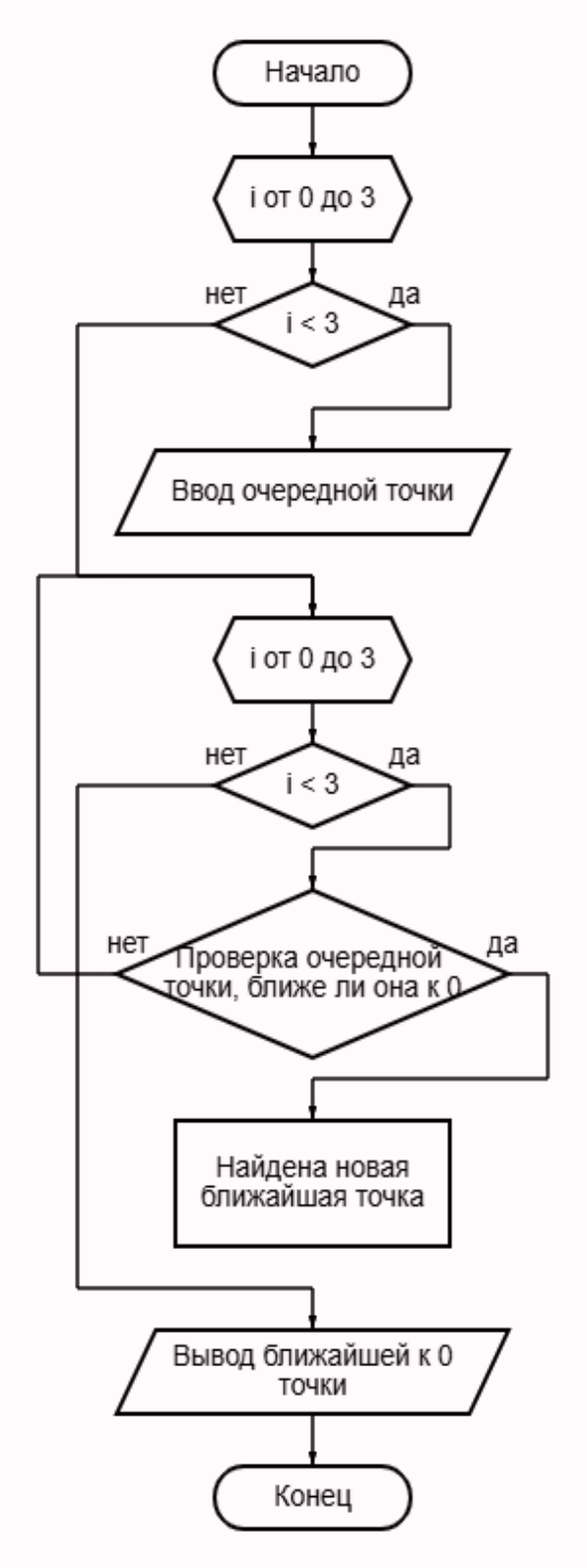
Листинг программного кода

ЗАДАНИЕ 2. ЧАСТЬ 1

ВАРИАНТ 11

На оси OX расположены три точки x1 > 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;
}
```

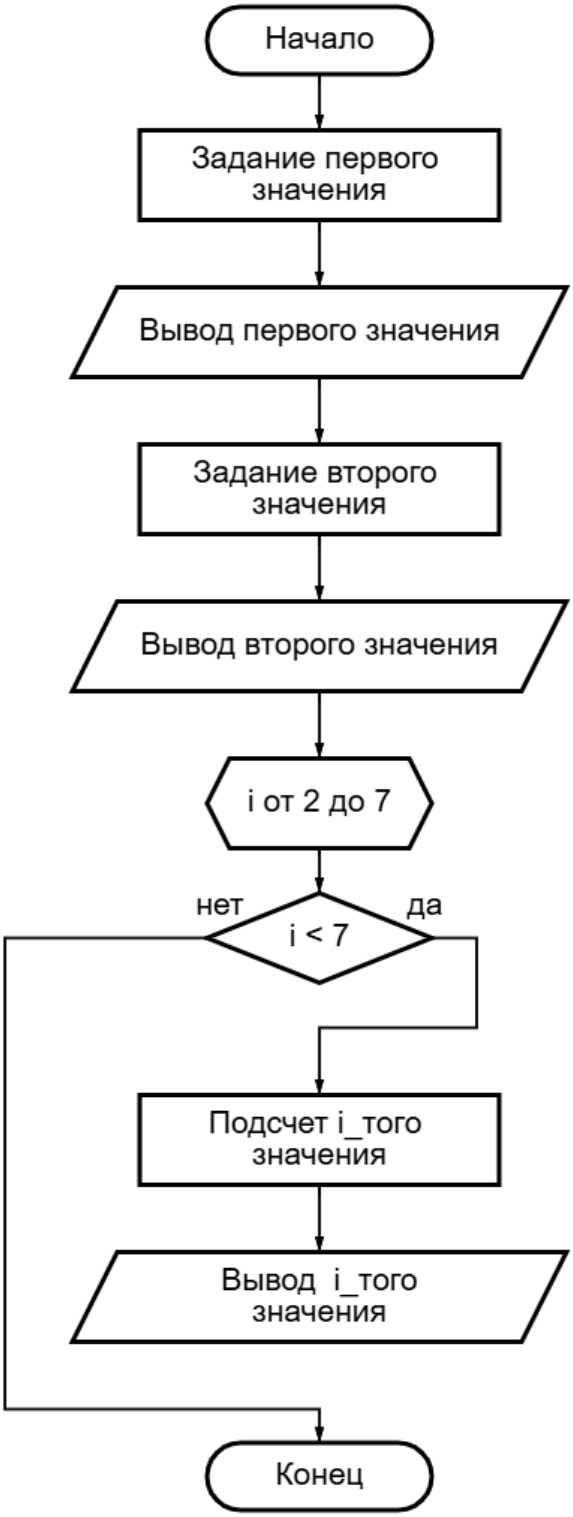
ЗАДАНИЕ 2. ЧАСТЬ 2

ВАРИАНТ 12

Сформируйте массив из значений полинома Лаггера

$L_0(x)=1, L_1(x)=x-1, L_n=(x-2n+1)L_{(n-1)}(x)-(n-1)^2 L_{(n-2)}(x)$ 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

Формула: $Z = \sin\left(\sum_{K=3}^{10} Y_K\right) + B \cdot \cos\left(\sum_{K=6}^{20} Y_K\right) + \frac{C}{\sum_{K=11}^{30} Y_K}$;

функция: $Y_K = b \cdot \frac{\ln(10 \cdot (A \cdot K + C))}{\sqrt{K + A + B}}$,

где $b = 1, A = 0, B = 9, C = 1$.

Расчет сумм в формуле и расчет Y_K должны быть оформлены в виде отдельных функций.

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 6x6. 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 6x6. 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;
}
```