

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»  
(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем

«УЧЕБНАЯ ОЗНАКОМИТЕЛЬНАЯ ПРАКТИКА»

тема: КОМПЬЮТЕРНАЯ ПРАКТИКА

Автор работы \_\_\_\_\_ Аладиб Язан ПВ-202

Руководитель работы \_\_\_\_\_ Гаврющенко Александр Павлович

Оценка \_\_\_\_\_

Белгород  
2022 г.

# Оглавление

1	ТЕМА 1. ЛИНЕЙНЫЕ АЛГОРИТМЫ	3
1.1	Задания варианта №2: . . . . .	3
1.2	Описание подпрограмм: . . . . .	3
1.3	Тестовые данные: . . . . .	3
1.4	Текст программы: . . . . .	4
2	ТЕМА 2. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ	5
2.1	Задания варианта №2: . . . . .	5
2.2	Описание подпрограмм: . . . . .	5
2.3	Тестовые данные: . . . . .	5
2.4	Текст программы: . . . . .	6
3	ТЕМА 3. ЦИКЛИЧЕСКИЕ И ИТЕРАЦИОННЫЕ АЛГОРИТМЫ	7
3.1	Задания варианта №2: . . . . .	7
3.2	Описание подпрограмм: . . . . .	7
3.3	Тестовые данные: . . . . .	7
3.4	Текст программы: . . . . .	8
4	ТЕМА 4. ПРОСТЕЙШИЕ ОПЕРАЦИИ НАД МАССИВАМИ	9
4.1	Задания варианта №2: . . . . .	9
4.2	Описание подпрограмм: . . . . .	9
4.3	Тестовые данные: . . . . .	10
4.4	Текст программы: . . . . .	10
5	ТЕМА 5. ВЕКТОРЫ И МАТРИЦЫ	12
5.1	Задания варианта №2: . . . . .	12
5.2	Описание подпрограмм: . . . . .	12
5.3	Тестовые данные: . . . . .	13
5.4	Текст программы: . . . . .	13
6	ТЕМА 6. ЛИНЕЙНЫЙ ПОИСК	15
6.1	Задания варианта №2: . . . . .	15
6.2	Описание подпрограмм: . . . . .	15
6.3	Тестовые данные: . . . . .	16
6.4	Текст программы: . . . . .	16

7	ТЕМА 7. АРИФМЕТИКА	19
7.1	Задания варианта №2: . . . . .	19
7.2	Описание подпрограмм: . . . . .	19
7.3	Тестовые данные: . . . . .	20
7.4	Текст программы: . . . . .	20
8	ТЕМА 8.ГЕОМЕТРИЯ И ТЕОРИЯ МНОЖЕСТВ	22
8.1	Задания варианта №2: . . . . .	22
8.2	Описание подпрограмм: . . . . .	22
8.3	Тестовые данные: . . . . .	23
8.4	Текст программы: . . . . .	23
9	ТЕМА 9. ЛИНЕЙНАЯ АЛГЕБРА И СЖАТИЕ ИНФОРМАЦИИ	27
9.1	Задания варианта №2: . . . . .	27
9.2	Описание подпрограмм: . . . . .	27
9.3	Тестовые данные: . . . . .	28
9.4	Текст программы: . . . . .	28
10	ТЕМА 10. АЛГОРИТМЫ ОБРАБОТКИ СИМВОЛЬНОЙ ИНФОРМАЦИИ	31
10.1	Задания варианта №2: . . . . .	31
10.2	Описание подпрограмм: . . . . .	31
10.3	Текст программы: . . . . .	32
11	ТЕМА 11. АНАЛИТИЧЕСКАЯ ГЕОМЕТРИЯ	34
11.1	Выполнение работы: . . . . .	34
12	ТЕМА 12. КРИВЫЕ ВТОРОГО ПОРЯДКА НА ПЛОСКОСТИ	35
12.1	Выполнение работы: . . . . .	35
13	ТЕМА 13. ГРАФИЧЕСКОЕ РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ	36
13.1	Выполнение работы: . . . . .	36
14	ТЕМА 14. ПЛОСКОСТЬ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ	37
14.1	Выполнение работы: . . . . .	37
15	ТЕМА 15. ПОВЕРХНОСТЬ ВТОРОГО ПОРЯДКА В ТРЕХ МЕРНОМ ПРОСТРАНСТВЕ	38
15.1	Выполнение работы: . . . . .	38

## Глава 1

# ТЕМА 1. ЛИНЕЙНЫЕ АЛГОРИТМЫ

### 1.1 Задания варианта №2:

Угол  $\alpha$  задан в радианах. Найти его величину в градусах, минутах и секундах.

### 1.2 Описание подпрограмм:

процедура (rad\_to\_deg):

Спецификация:

1. Заголовок: rad\_to\_deg(double rad)
2. Назначение: перевести из радианов в градусы

### 1.3 Тестовые данные:

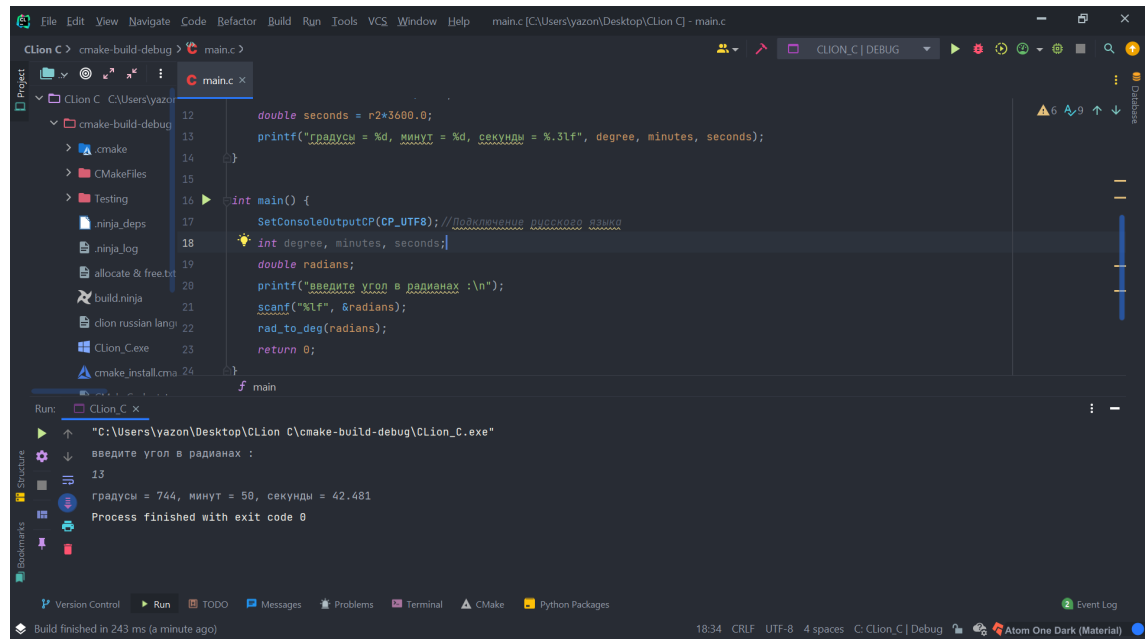
Исходные данные	Результаты
5	градусы = 286, минут = 28, секунды = 44.031
20	градусы = 1145, минут = 54, секунды = 56.125
13	градусы = 744, минут = 50, секунды = 42.481

## 1.4 Текст программы:

```
#include <stdio.h>
#include <math.h>
#include <windows.h>

void rad_to_deg(double rad){
    double radians = rad*180/M_PI ; //M_PI = 3.141592
    int degree = radians;
    double r1 = radians-degree;
    int minutes = r1 * 60.0;
    double r2 = r1 - minutes/60.0;
    double seconds = r2*3600.0;
    printf("градусы = %d, минут = %d, секунды = %.3lf"
    ,degree, minutes, seconds);
}

int main() {
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    int degree, minutes, seconds;
    double radians;
    printf("введите угол в радианах :\n");
    scanf("%lf", &radians);
    rad_to_deg(radians);
    return 0;
}
```



The screenshot shows the CLion IDE interface. The main editor displays the C source code from the previous block. The left sidebar shows the project structure with folders like 'cmake-build-debug' and 'CLion\_C'. The bottom panel is split into two sections: 'Run' and 'Output'. The 'Run' section shows the command to execute the program: `"C:\Users\yazon\Desktop\CLion_C\cmake-build-debug\CLion_C.exe"`. The 'Output' section shows the program's execution results: `введите угол в радианах :` followed by a prompt, and then the output `градусы = 744, минут = 50, секунды = 42.481`. The status bar at the bottom indicates the build finished in 243 ms and the current encoding is UTF-8.

## Глава 2

# ТЕМА 2. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

### 2.1 Задания варианта №2:

Треугольник задан длинами своих сторон:  $a, b, c$ . Определить, является ли он тупоугольным, прямоугольным или остроугольным.

### 2.2 Описание подпрограмм:

процедура (check):

Спецификация:

1. Заголовок: `check(int a, int b, int c)`

2. Назначение: проверить, является ли треугольник остроугольным, прямоугольным или тупоугольным

### 2.3 Тестовые данные:

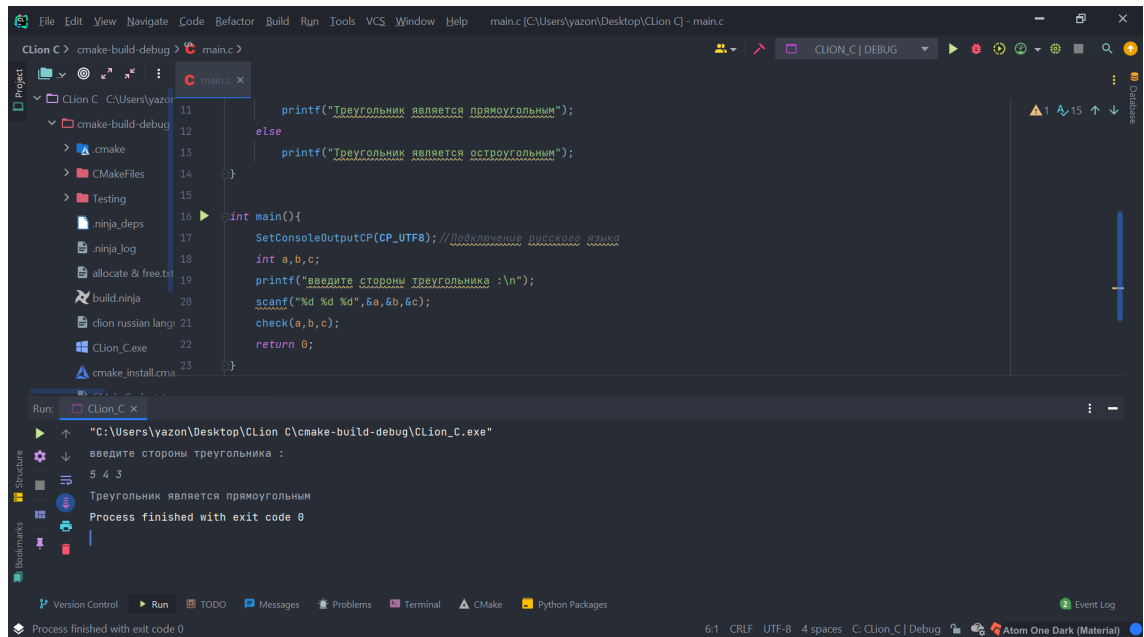
Исходные данные			Результаты
10	5	2	Треугольник является тупоугольным
10	7	8	Треугольник является остроугольным
5	4	3	Треугольник является прямоугольным

## 2.4 Текст программы:

```
#include <stdio.h>
#include <windows.h>

void check(int a,int b,int c){
    double A = b * b + c * c - a * a;
    double B = c * c + a * a - b * b;
    double C = a * a + b * b - c * c;
    if (A < 0 || B < 0 || C < 0)
        printf("Треугольник является тупоугольным");
    else if (A == 0 || B == 0 || C == 0)
        printf("Треугольник является прямоугольным");
    else
        printf("Треугольник является остроугольным");
}

int main(){
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    int a,b,c;
    printf("введите стороны треугольника :\n");
    scanf("%d %d %d",&a,&b,&c);
    check(a,b,c);
    return 0;
}
```



## Глава 3

# ТЕМА 3. ЦИКЛИЧЕСКИЕ И ИТЕРАЦИОННЫЕ АЛГОРИТМЫ

### 3.1 Задания варианта №2:

Для заданного  $\varepsilon$  найти наименьшее  $n$  такое, что  $\frac{2^n}{n!} < \varepsilon$ . Вывести все члены последовательности от 1-го до  $n$ -го.

### 3.2 Описание подпрограмм:

функция (sequence):

Спецификация:

1. Заголовок: sequence(int n)
2. Назначение: Вычислите и распечатайте последовательность

### 3.3 Тестовые данные:

Исходные данные	Результаты					
$\varepsilon = 0.001$	1) 2	2) 2	3) 1.33333	4) 0.666667	5) 0.266667	6) 0.0888889
	7) 0.0253968	8) 0.00634921	9) 0.00141093	10) 0.000282187	n = 10	
$\varepsilon = 1$	1) 2	2) 2	3) 1.33333	4) 0.666667	n = 4	
$\varepsilon = 0.00001$	1) 2	2) 2	3) 1.33333	4) 0.666667	5) 0.266667	6) 0.0888889
	7) 0.0253968	8) 0.00634921	9) 0.00141093	10) 0.000282187		
	11) 5.13067e-05	12) 8.55112e-06	n = 12			

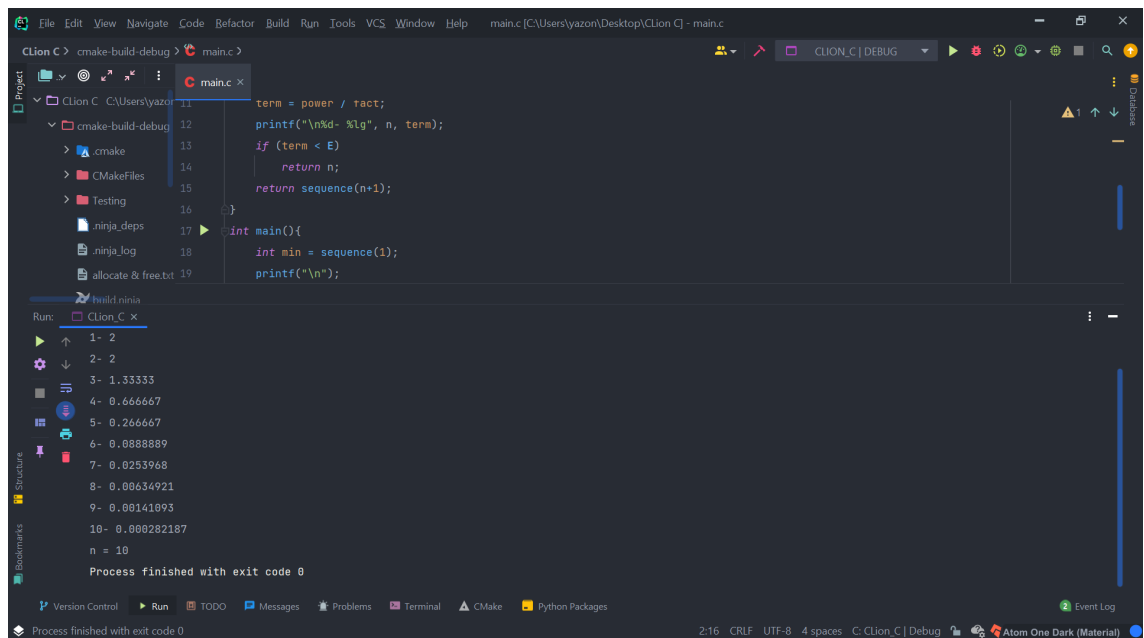


### 3.4 Текст программы:

```
#include <stdio.h>
#define E 0.00001

int sequence(int n){
    int i;
    double power = 1.0, fact = 1.0, term;
    for(i = 1; i <= n; i++) {
        power *= 2;
        fact *= i;
    }
    term = power / fact;
    printf("\n%- %lg", n, term);
    if (term < E)
        return n;
    return sequence(n+1);
}

int main(){
    int min = sequence(1);
    printf("\n");
    printf("n = %d", min);
    return 0;
}
```



The screenshot shows the CLion IDE interface. The main editor window displays the C++ code for the 'sequence' program. The Run window at the bottom shows the output of the program, which is a list of values for 'n' and 'term'.

Run: CLion\_C x

```
1- 2
2- 2
3- 1.33333
4- 0.666667
5- 0.266667
6- 0.0888889
7- 0.0253968
8- 0.00634921
9- 0.00141093
10- 0.000282187
n = 10
Process finished with exit code 0
```

## Глава 4

# ТЕМА 4. ПРОСТЕЙШИЕ ОПЕРАЦИИ НАД МАССИВАМИ

### 4.1 Задания варианта №2:

Элементы одномерного массива  $A(n^2)$  построчно расположить в матрице  $B(n,n)$ .

### 4.2 Описание подпрограмм:

процедура (input):

Спецификация:

1. Заголовок: `input(int array[N*N])`
2. Назначение: Ввод массива (array) размера  $N*N$  ( $N=3$ )

процедура (sort):

Спецификация:

1. Заголовок: `sort(int array[N*N],int array2[N][N])`
2. Назначение: сортировка элементов одномерного массива (array) построчно в матрице (array2).

процедура (output):

Спецификация:

1. Заголовок: `output(int array2[N][N])`
2. Назначение: распечатать матрицу

### 4.3 Тестовые данные:

Исходные данные	Результаты
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
11 22 33 44 55 66 77 88 99	11 22 33 44 55 66 77 88 99

### 4.4 Текст программы:

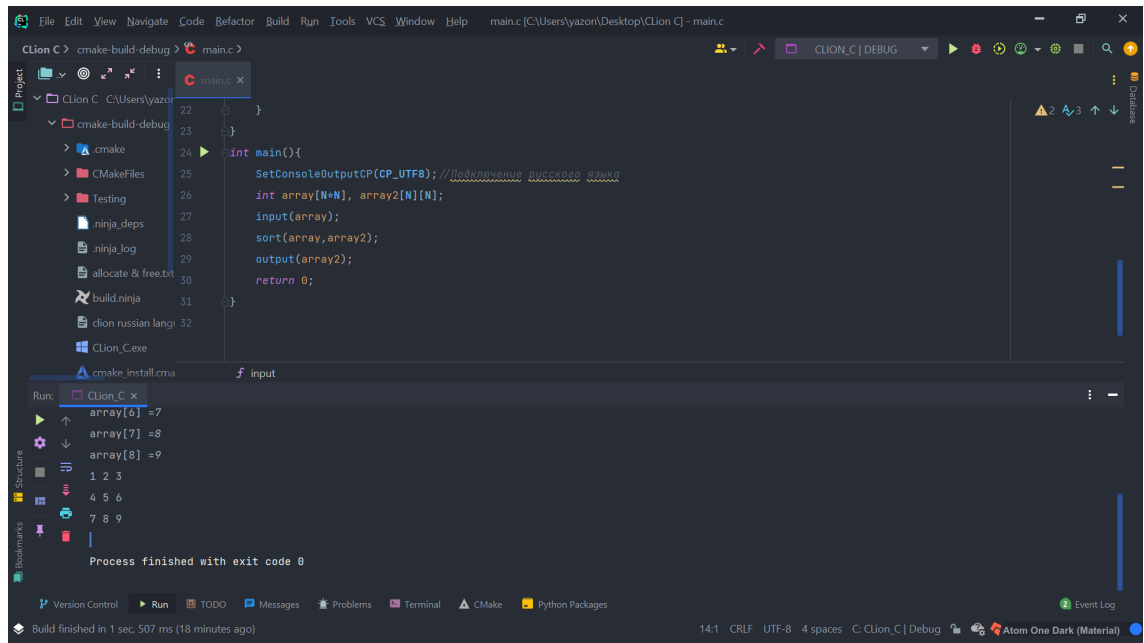
```
#include <stdio.h>
#include <windows.h>
#define N 3

void input(int array[N*N]){
    for(int i=0; i<N*N; i++){
        printf("array[%d] =",i);
        scanf("%d",&array[i]);
    }
}

void sort(int array[N*N],int array2[N][N]){
    for(int i=0; i<N; i++)
        for(int j=0; j<N; j++)
            array2[i][j] = array[i*N+j];
}

void output(int array2[N][N]){
    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++) {
            printf("%d ", array2[i][j]);
        }
        printf("\n");
    }
}

int main(){
    SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
    int array[N*N], array2[N][N];
    input(array);
    sort(array,array2);
    output(array2);
    return 0;
}
```



## Глава 5

# ТЕМА 5. ВЕКТОРЫ И МАТРИЦЫ

### 5.1 Задания варианта №2:

Матрицу  $M(m,n)$  заполнить натуральными числами от 1 до  $m \cdot n$  по спирали, начинающейся в левом верхнем углу и закрученной по часовой стрелке.

### 5.2 Описание подпрограмм:

процедура (sort):

Спецификация:

1. Заголовок: `sort(int m, int n, int M[m][n])`
2. Назначение: заполнить и отсортировать матрицу по часовой стрелке

процедура (output):

Спецификация:

1. Заголовок: `output(int m, int n, int M[m][n])`
2. Назначение: распечатать матрицу

### 5.3 Тестовые данные:

Исходные данные	Результаты
m = 3      n = 3	1 2 3 8 9 4 7 6 5
m = 5      n = 3	1 2 3 12 13 4 11 14 5 10 15 6 9 8 7
m = 2      n = 5	1 2 3 4 5 10 9 8 7 6

### 5.4 Текст программы:

```
#include <stdio.h>
#include <windows.h>

void sort(int m, int n, int M[m][n]){
    int val = 1;
    int k = 0, l = 0;
    while (k < m && l < n){
        for (int i = l; i < n; ++i)
            M[k][i] = val++;
        k++;
        for (int i = k; i < m; ++i)
            M[i][n-1] = val++;
        n--;
        if (k < m){
            for (int i = n-1; i >= l; --i)
                M[m-1][i] = val++;
            m--;
        }
        if (l < n){
            for (int i = m-1; i >= k; --i)
                M[i][l] = val++;
            l++;
        }
    }
}

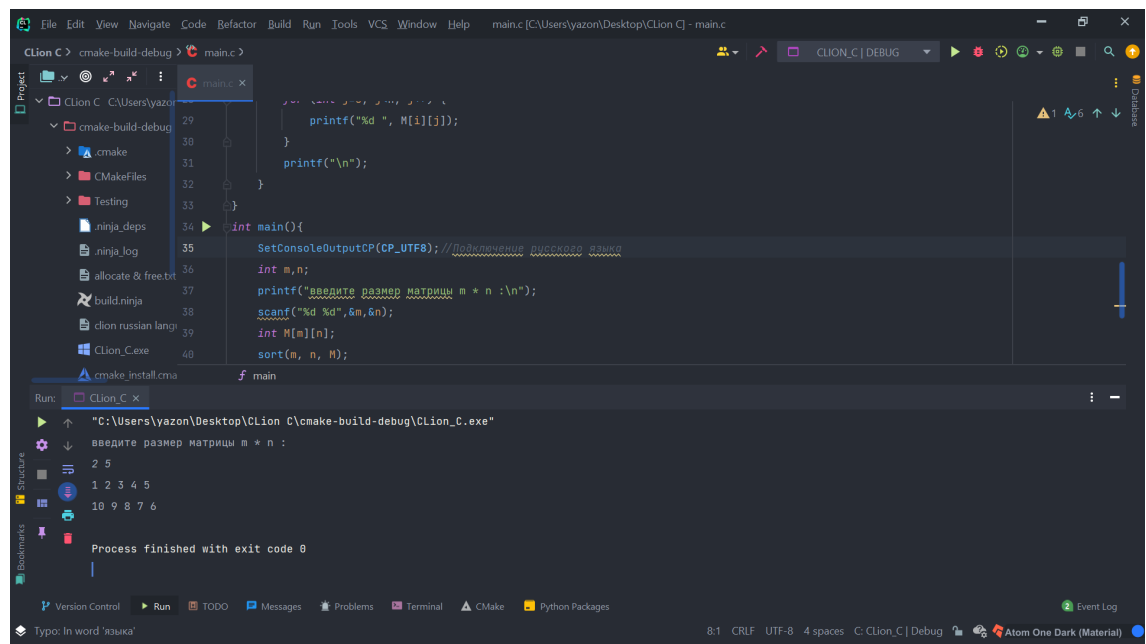
void output(int m, int n, int M[m][n]){
    for (int i=0; i<m; i++){
```

```

        for (int j=0; j<n; j++) {
            printf("%d ", M[i][j]);
        }
        printf("\n");
    }
}

int main(){
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    int m,n;
    printf("введите размер матрицы m * n :\n");
    scanf("%d %d",&m,&n);
    int M[m][n];
    sort(m, n, M);
    output(m,n,M);
    return 0;
}

```



## Глава 6

# ТЕМА 6. ЛИНЕЙНЫЙ ПОИСК

### 6.1 Задания варианта №2:

Седловой точкой в матрице называется элемент, являющийся одновременно наибольшим в столбце и наименьшим в строке. Седловых точек может быть несколько. В матрице  $A(m,n)$  найти все седловые точки либо установить, что таких точек нет.

### 6.2 Описание подпрограмм:

процедура (input):

Спецификация:

1. Заголовок: `input(int m,int n,int A[m][n])`
2. Назначение: Ввод массива (A) размера  $m*n$

процедура (output):

Спецификация:

1. Заголовок: `output(int m,int n,int A[m][n])`
2. Назначение: распечатать матрицу

процедура (findsaddlepoints):

Спецификация:

1. Заголовок: `findsaddlepoints(int m,int n,int A[m][n])`
2. Назначение: нахождение седловых точек в матрице



### 6.3 Тестовые данные:

Исходные данные	Результаты
$m = 3 \quad n = 3$ элементы матрицы : 1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9 седловая точка: A[3][1]=7
$m = 5 \quad n = 3$ элементы матрицы : 7 8 19 23 53 74 12 32 54 16 31 99 10 17 88	7 8 19 23 53 74 12 32 54 16 31 99 10 17 88 Матрица не имеет седловых точек!

### 6.4 Текст программы:

```

#include <stdio.h>
#include <windows.h>

void input(int m,int n,int A[m][n]){
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%d",&A[i][j]);
        }
    }
}

void output(int m,int n,int A[m][n]){
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            printf("%d ",A[i][j]);
        }
        printf("\n");
    }
}

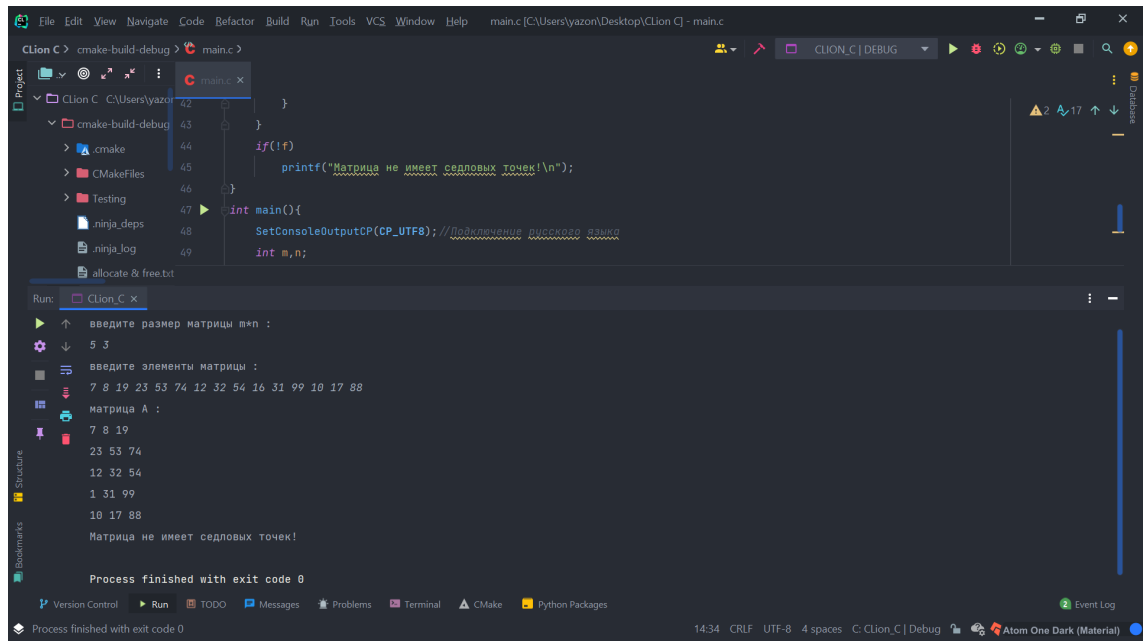
void findsaddlepoints(int m,int n,int A[m][n]){
    int min,max,f=0;
    for (int i=0;i<n;i++){
        min=A[i][0];
        for (int x=1;x<m;x++){
            if (A[i][x]<min){
                min=A[i][x];
            }
        }
    }
}

```

```

    }
}
for(int j=0;j<n;j++){
    if (A[i][j]==min){
        max=A[0][j];
        for (int y=1;y<n;y++){
            if (A[y][j]>max){
                max= A[y][j];
            }
        }
        if (A[i][j]==max){
            printf("седловая точка: \n");
            printf( "A[%d] [%d]=%d\n", i+1, j+1, A[i][j] );
            f=1;
        }
    }
}
}
if(!f)
    printf("Матрица не имеет седловых точек!\n");
}
int main(){
    SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
    int m,n;
    printf("введите размер матрицы m*n :\n");
    scanf("%d %d",&m,&n);
    int A[m][n];
    printf("введите элементы матрицы :\n");
    input(m,n,A);
    output(m,n,A);
    findsaddlepoints(m,n,A);
    return 0;
}

```



## Глава 7

# ТЕМА 7. АРИФМЕТИКА

### 7.1 Задания варианта №2:

Для натуральных чисел, не превосходящих заданного  $k$ , проверить признак делимости на 9 (сумма цифр числа, делящегося на 9, также делится на 9). Вывести  $m$  последних таких чисел ( $m \ll k$ ).

### 7.2 Описание подпрограмм:

функция (Sum):

Спецификация:

1. Заголовок: Sum(int n)
2. Назначение: вернуть сумму цифр числа

функция (divisibility):

Спецификация:

1. Заголовок: divisibility (int value)
2. Назначение: проверить, может ли число делиться на 9

процедура (output):

Спецификация:

1. Заголовок: output(int k, int m)
2. Назначение: Вывести числа, которые делятся на 9

### 7.3 Тестовые данные:

Исходные данные	Результаты
k = 100      m = 11	числа, которые могут делиться на 9 : 99 90 81      72 63 54      45 36 27 18 9
k = 100      m = 4	числа, которые могут делиться на 9 : 99 90 81      72

### 7.4 Текст программы:

```
#include <stdio.h>
#include <windows.h>

int Sum(int n){
    int sum = 0;
    while (n != 0){
        sum += n % 10;
        n /= 10;
    }
    return sum;
}

int divisibility (int value){
    int digits_sum = Sum(value);
    if (digits_sum % 9 ==0)
        return 1;
    return 0;
}

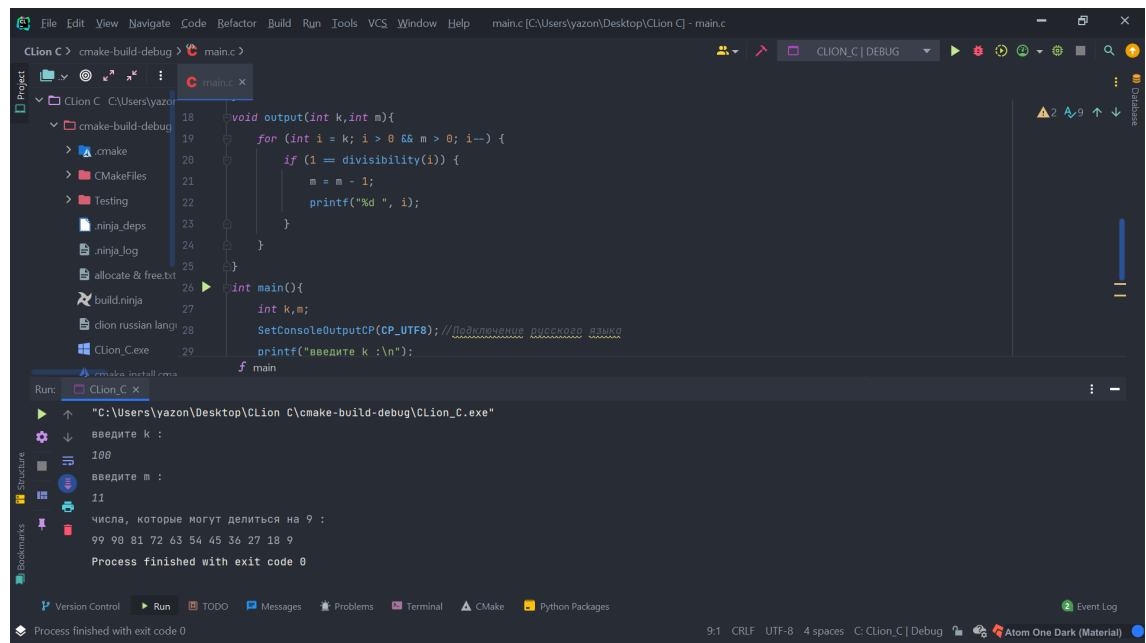
void output(int k,int m){
    for (int i = k; i > 0 && m > 0; i--) {
        if (1 == divisibility(i)) {
            m = m - 1;
            printf("%d ", i);
        }
    }
}

int main(){
    int k,m;
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    printf("введите k :\n");
    scanf("%d", &k);
    printf("введите m :\n");
```

```

scanf("%d", &m);
printf("числа, которые могут делиться на 9 :\n");
output(k,m);
return 0;
}

```



## Глава 8

# ТЕМА 8.ГЕОМЕТРИЯ И ТЕОРИЯ МНОЖЕСТВ

### 8.1 Задания варианта №2:

Задано множество точек на плоскости. Найти выпуклую оболочку этого множества, то есть выпуклый многоугольник с вершинами в некоторых точках этого множества, охватывающий все его точки.

### 8.2 Описание подпрограмм:

функция (ccw):

Спецификация:

1. Заголовок: `ccw(const Point *a, const Point *b, const Point *c)`
2. Назначение: найти ориентацию упорядоченной тройки

функция (comparePoints):

Спецификация:

1. Заголовок: `comparePoints(const void *lhs, const void *rhs)`
2. Назначение: Выводит выпуклую оболочку набора точек.

процедура (xmalloc):

Спецификация:

1. Заголовок: `xmalloc(size_t n)`
2. Назначение: выделение памяти

процедура (xrealloc):

Спецификация:

1. Заголовок: xrealloc(void\* p, size\_t n)
2. Назначение: перераспределение

процедура (printPoints):

Спецификация:

1. Заголовок: printPoints(const Point\* points, int len)
2. Назначение: распечатать точки (результат)

функция (convexHull):

Спецификация:

1. Заголовок: convexHull(Point p[], int len, int\* hsize)
2. Назначение: найти выпуклую оболочку множества точек.

### 8.3 Тестовые данные:

Исходные данные	Результаты
16,3] [12,17] [0,6] [-4,-6] [16,6] [16,-7] [16,-3] [17,-4] [5,19] [19,-8] [3,16] [12,13] [3,-4] [17,5] [-3,15] [-3,-9] [0,11] [-9,-3] [-4,-2] [12,10]	Выпуклая оболочка: [(-9,-3),(-3,-9),(19,-8) (17,5),(12,17),(5,19),(-3,15)]

### 8.4 Текст программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <windows.h>

typedef struct tPoint{
    int x, y;
} Point;

int ccw(const Point *a, const Point *b, const Point *c){
    return (b->x - a->x) * (c->y - a->y)
        > (b->y - a->y) * (c->x - a->x);
}

int comparePoints(const void *lhs, const void *rhs){
```



```

    const Point* lp = lhs;
    const Point* rp = rhs;
    if (lp->x < rp->x)
        return -1;
    if (rp->x < lp->x)
        return 1;
    if (lp->y < rp->y)
        return -1;
    if (rp->y < lp->y)
        return 1;
    return 0;
}

void* xmalloc(size_t n){
    void* ptr = malloc(n);
    return ptr;
}

void* xrealloc(void* p, size_t n){
    void* ptr = realloc(p, n);
    return ptr;
}

void printPoints(const Point* points, int len){
    printf("[");
    if (len > 0) {
        const Point* ptr = points;
        const Point* end = points + len;
        printf("(%d, %d)", ptr->x, ptr->y);
        ++ptr;
        for (; ptr < end; ++ptr)
            printf(", (%d, %d)", ptr->x, ptr->y);
    }
    printf("]");
}

Point* convexHull(Point p[], int len, int* hsize){
    if (len == 0) {
        *hsize = 0;
        return NULL;
    }
    int i, size = 0, capacity = 4;
    Point* hull = xmalloc(capacity * sizeof(Point));
    qsort(p, len, sizeof(Point), comparePoints);
    for (i = 0; i < len; ++i) {
        while (size >= 2 && !ccw(&hull[size - 2], &hull[size - 1], &p[i]))
            --size;
        if (size == capacity) {
            capacity *= 2;
            hull = xrealloc(hull, capacity * sizeof(Point));
        }
        hull[size++] = p[i];
    }
    *hsize = size;
    return hull;
}

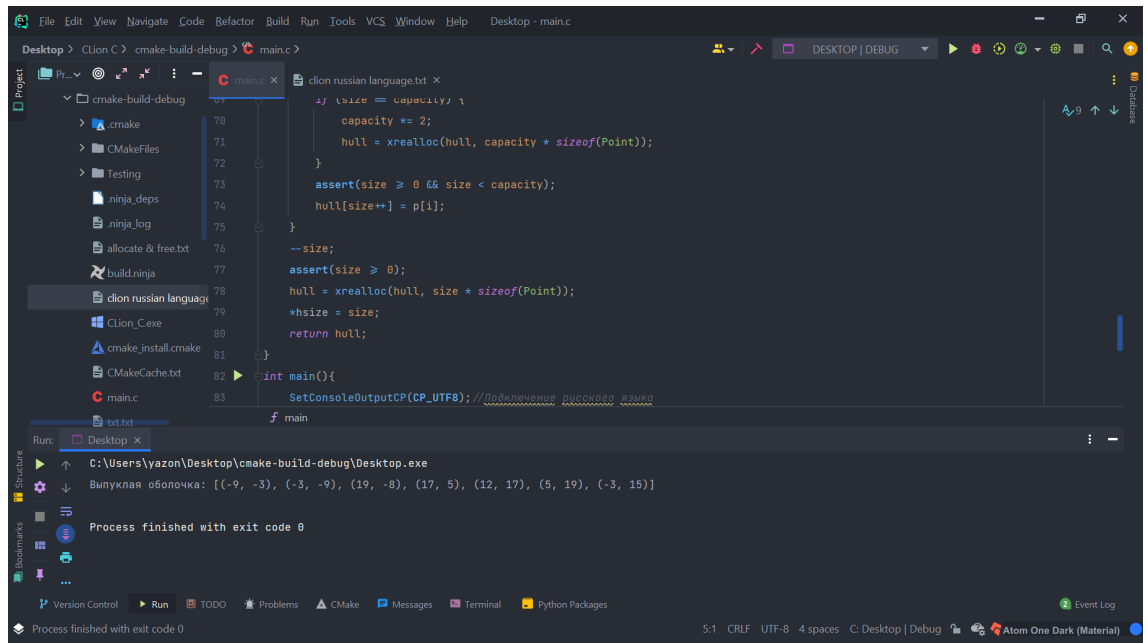
```

```

    }
    assert(size >= 0 && size < capacity);
    hull[size++] = p[i];
}
int t = size + 1;
for (i = len - 1; i >= 0; i--) {
    while (size >= t && !ccw(&hull[size - 2], &hull[size - 1], &p[i]))
        --size;
    if (size == capacity) {
        capacity *= 2;
        hull = xrealloc(hull, capacity * sizeof(Point));
    }
    assert(size >= 0 && size < capacity);
    hull[size++] = p[i];
}
--size;
assert(size >= 0);
hull = xrealloc(hull, size * sizeof(Point));
*hsize = size;
return hull;
}

int main(){
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    Point points[] = {
        {16, 3}, {12, 17}, {0, 6}, {-4, -6}, {16, 6},
        {16, -7}, {16, -3}, {17, -4}, {5, 19}, {19, -8},
        {3, 16}, {12, 13}, {3, -4}, {17, 5}, {-3, 15},
        {-3, -9}, {0, 11}, {-9, -3}, {-4, -2}, {12, 10}
    };
    int hsize;
    Point* hull = convexHull(points, sizeof(points)/sizeof(Point), &hsize);
    printf("Выпуклая оболочка: ");
    printPoints(hull, hsize);
    printf("\n");
    free(hull);
    return 0;
}

```



## Глава 9

# ТЕМА 9. ЛИНЕЙНАЯ АЛГЕБРА И СЖАТИЕ ИНФОРМАЦИИ

### 9.1 Задания варианта №2:

Выполнить операцию транспонирования прямоугольной матрицы  $A(m,n)$ ,  $m!=n$ , не выделяя дополнительномассива для хранения результата. Матрицу представить в виде одномерного массива.

### 9.2 Описание подпрограмм:

процедура (input):

Спецификация:

1. Заголовок: input(int m,int n,int \*\*A)
2. Назначение: Ввод массива (A) размера  $m*n$

процедура (original):

Спецификация:

1. Заголовок: original(int m,int n,int \*\*A)
2. Назначение: распечатать матрицу перед транспозицией

процедура (transposition):

Спецификация:

1. Заголовок: transposition(int m,int n,int \*\*A)
2. Назначение: распечатать матрицу транспонирования

### 9.3 Тестовые данные:

Исходные данные	Результаты
<p>m = 3      n = 2</p> <p>элементы матрицы:</p> <p>1 2 3 4 5 6</p>	<p>Полученная матрица :</p> <p>1 2</p> <p>3 4</p> <p>5 6</p> <p>Результат транспонирования:</p> <p>1 3 5</p> <p>2 4 6</p>
<p>m = 2      n =</p> <p>элементы матрицы:</p> <p>16 23 65 11 98 34 59 72</p>	<p>Полученная матрица :</p> <p>16 23 65 11</p> <p>98 34 59 7</p> <p>Результат транспонирования:</p> <p>16 98</p> <p>23 34</p> <p>65 59</p> <p>11 72</p>

### 9.4 Текст программы:

```
#include <stdio.h>
#include <windows.h>

void input(int m,int n,int **A){
    for(int i=0;i<m;i++){
        for (int j=0;j<n;j++){
            printf("A[%d] [%d]=", i, j);
            scanf("%d", &A[i][j]);
        }
    }
}

void original(int m,int n,int **A){
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            printf("%d ",A[i][j]);
        }
        printf("\n");
    }
}

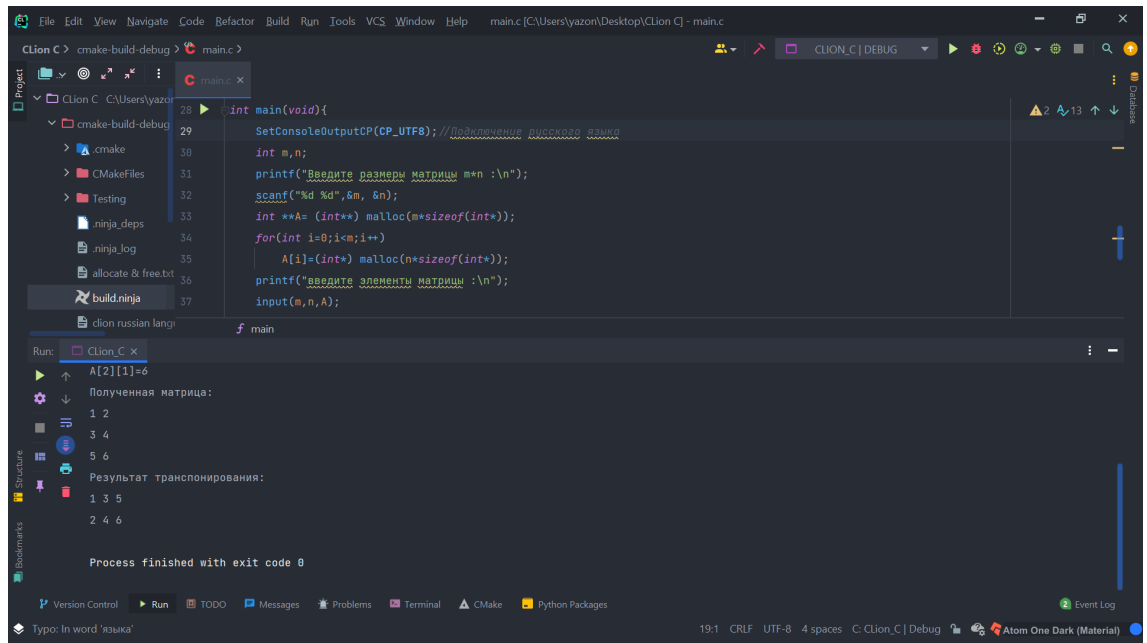
void transposition(int m,int n,int **A){
```

```

        for(int i=0;i<n;i++) {
            for (int j=0;j<m;j++) {
                printf("%d ", A[j][i]);
            }
            printf("\n");
        }
    }
}

int main(void){
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    int m,n;
    printf("Введите размеры матрицы m*n :\n");
    scanf("%d %d",&m, &n);
    int **A= (int**) malloc(m*sizeof(int*));
    for(int i=0;i<m;i++)
        A[i]=(int*) malloc(n*sizeof(int*));
    printf("введите элементы матрицы :\n");
    input(m,n,A);
    printf("Полученная матрица :\n");
    original(m,n,A);
    printf("Результат транспонирования :\n");
    transposition(m,n,A);
    for(int i=0;i<m;i++)
        free(A[i]);
    free(A);
    return 0;
}

```



## Глава 10

# ТЕМА 10. АЛГОРИТМЫ ОБРАБОТКИ СИМВОЛЬНОЙ ИНФОРМАЦИИ

### 10.1 Задания варианта №2:

В заданном тексте найти самое длинное слово и самую длинную фразу.

### 10.2 Описание подпрограмм:

процедура (longestPhrase):

Спецификация:

1. Заголовок: longestPhrase(char\* text)
2. Назначение: найти самую длинную фразу

процедура (longestWord):

Спецификация:

1. Заголовок: longestWord(char\* text)
2. Назначение: найти самое длинное слово



### 10.3 Текст программы:

```
#include <stdio.h>
#include <windows.h>

void longestPhrase(char* text) {
    int max = 0;
    char Punctuation[] = ".,-?:_!";
    char *first = text, *last, *firstMax = text, *lastMax;
    for( ;*text; text++) {
        for(int i = 0; Punctuation[i]; i++) {
            if (*text == Punctuation[i]) {
                last = text;
                if (last - first > max) {
                    max = last - first;
                    firstMax = first;
                    lastMax = last;
                }
                first = text + 1;
                break;
            }
        }
    }
    printf("самая длинная фраза:\n");
    while(firstMax <= lastMax)
        putchar(*firstMax++);
    putchar('\n');
}

void longestWord(char* text){
    int i, index=0, max=0, count=0, len = strlen(text);
    for (i=0; i<len; i++)
        if (text[i] != ' ')
            count ++;
        else {
            if (count > max) {
                max = count;
                index = i - count;
            }
            count = 0;
        }
    if (count > max) {
        max = count;
        index = i - count;
    }
    max += index;
    printf("Самое длинное слово:\n");
```

```

        for (i=index; i<max; i++)
            putchar(text[i]);
    }
    int main(){
        SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
        char text[] = "C is a general purpose computer programming"
            "language. It was created in the 1970s by"
            "Dennis Ritchie, and remains very widely used"
            "and influential. By design, C's features cleanly"
            "reflect the capabilities of the targeted CPUs."
            "It has found lasting use in operating systems,"
            "device drivers, protocol stacks, though decreasingly"
            "for application software, and is common in"
            "computer architectures that range from the largest"
            "supercomputers to the smallest microcontrollers"
            "and embedded systems."
        ;
        longestPhrase(text);
        printf("\n");
        longestWord(text);
        return 0;
    }

```

The screenshot shows a C++ IDE with the following components:

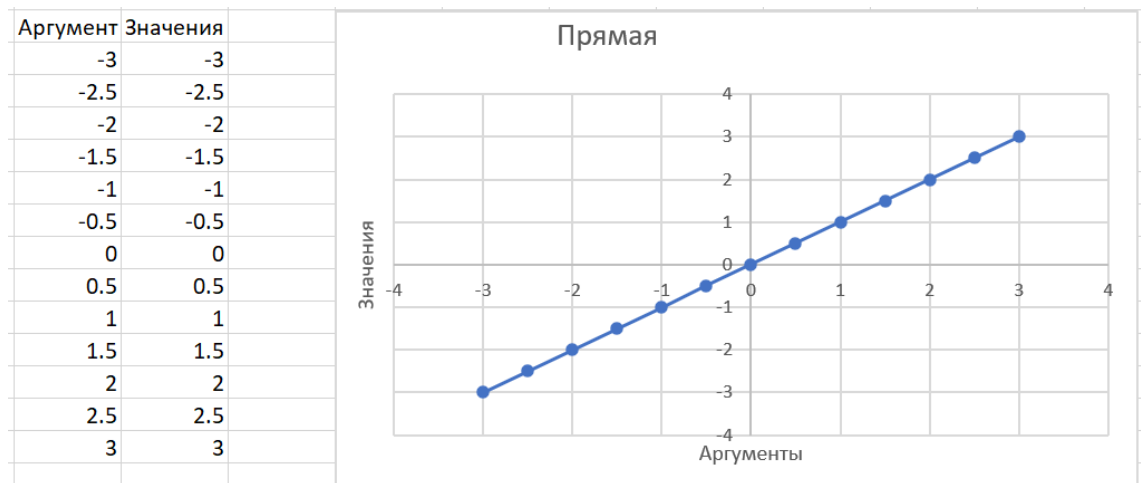
- Editor:** Displays the C++ code from the previous block. The text is in Russian, and the code uses UTF-8 encoding.
- Run Window:** Shows the output of the program. It displays the longest phrase and the longest word found in the input text.
- Project Explorer:** Shows the project structure, including files like 'main.c', 'clon russian language.txt', and 'txt.txt'.
- Terminal:** Shows the command prompt output, indicating the process finished with exit code 0.

## Глава 11

# ТЕМА 11. АНАЛИТИЧЕСКАЯ ГЕОМЕТРИЯ

### 11.1 Выполнение работы:

2: Построить биссектрису I—III координатных углов декартовой системы координат в диапазоне  $x \in [-3; 3]$  с шагом  $\Delta = 0,5$ .



## Глава 12

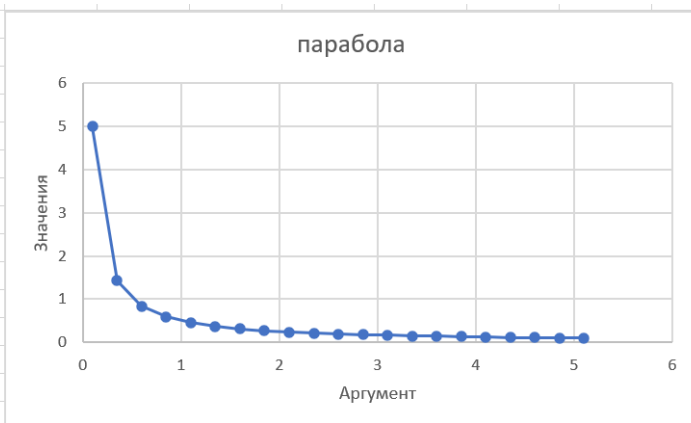
# ТЕМА 12. КРИВЫЕ ВТОРОГО ПОРЯДКА НА ПЛОСКОСТИ

### 12.1 Выполнение работы:

2 : Построить гиперболу при  $0,1 \leq x \leq 5,1$  с шагом  $\Delta = 0,25$ .

$$y = \frac{1}{2x}$$

Аргумент	Значения
0.1	5
0.35	1.428571
0.6	0.833333
0.85	0.588235
1.1	0.454545
1.35	0.37037
1.6	0.3125
1.85	0.27027
2.1	0.238095
2.35	0.212766
2.6	0.192308
2.85	0.175439
3.1	0.16129
3.35	0.149254
3.6	0.138889
3.85	0.12987
4.1	0.121951
4.35	0.114943
4.6	0.108696
4.85	0.103093
5.1	0.098039



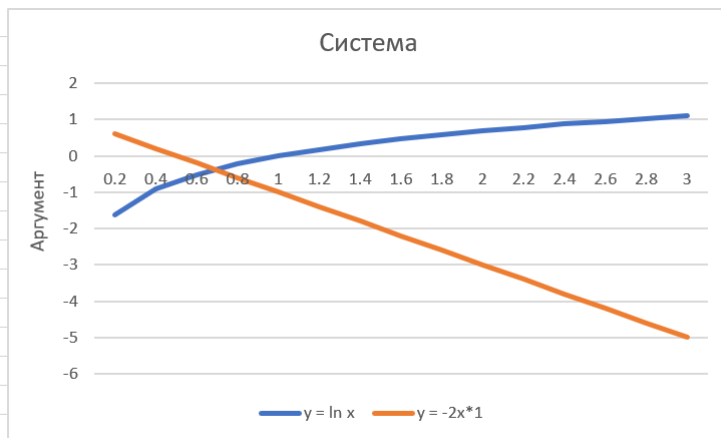
## Глава 13

# ТЕМА 13. ГРАФИЧЕСКОЕ РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ

### 13.1 Выполнение работы:

2. 
$$\begin{cases} y = \ln x \\ y = -2x + 1 \end{cases}$$
 в диапазоне  $0,2 \leq x \leq 2,5$ , с шагом  $\Delta = 0,2$ .

Аргумент	$y = \ln x$	$y = -2x + 1$
0.2	-1.60944	0.6
0.4	-0.91629	0.2
0.6	-0.51083	-0.2
0.8	-0.22314	-0.6
1	0	-1
1.2	0.182322	-1.4
1.4	0.336472	-1.8
1.6	0.470004	-2.2
1.8	0.587787	-2.6
2	0.693147	-3
2.2	0.788457	-3.4
2.4	0.875469	-3.8
2.6	0.955511	-4.2
2.8	1.029619	-4.6
3	1.098612	-5



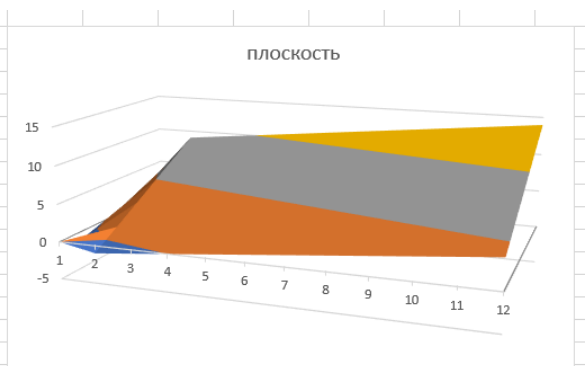
## Глава 14

# ТЕМА 14. ПЛОСКОСТЬ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ

### 14.1 Выполнение работы:

2. Построить плоскость, отсекающую на координатных осях отрезки  $a=3$ ,  $b=2$  и  $c=1$ , при  $-1 \leq x \leq 4$  с шагом  $\Delta=0.5$  и  $-1 \leq y \leq 3$  с шагом  $\Delta=1$ .

x\y	-1	0	1	2	3
-1	1	3	5	7	9
-0.5	1.5	3.5	5.5	7.5	9.5
0	2	4	6	8	10
0.5	2.5	4.5	6.5	8.5	10.5
1	3	5	7	9	11
1.5	3.5	5.5	7.5	9.5	11.5
2	4	6	8	10	12
2.5	4.5	6.5	8.5	10.5	12.5
3	5	7	9	11	13
3.5	5.5	7.5	9.5	11.5	13.5
4	6	8	10	12	14



## Глава 15

# ТЕМА 15. ПОВЕРХНОСТЬ ВТОРОГО ПОРЯДКА В ТРЕХ МЕРНОМ ПРОСТРАНСТВЕ

### 15.1 Выполнение работы:

2. Построить верхнюю часть гиперболоида, заданного уравнением

$$\frac{x^2}{9} + \frac{y^2}{4} - z^2 = -1, \text{ лежащую в диапазоне } -3 \leq x \leq 3, -2 \leq y \leq 2 \text{ с}$$

-1	-0.5	0	0.5	1	1.5	2
1.5	1.43614	1.41421	1.43614	1.5	1.60078	1.73205
1.39443	1.3255	1.30171	1.3255	1.39443	1.50231	1.64148
1.30171	1.22758	1.20185	1.22758	1.30171	1.41667	1.56347
1.22474	1.14564	1.11803	1.14564	1.22474	1.34629	1.5
1.16667	1.08333	1.05409	1.08333	1.16667	1.29368	1.45297
1.13039	1.04416	1.01379	1.04416	1.13039	1.26106	1.424
1.11803	1.03078	1	1.03078	1.11803	1.25	1.41421
1.13039	1.04416	1.01379	1.04416	1.13039	1.26106	1.424
1.16667	1.08333	1.05409	1.08333	1.16667	1.29368	1.45297
1.22474	1.14564	1.11803	1.14564	1.22474	1.34629	1.5
1.30171	1.22758	1.20185	1.22758	1.30171	1.41667	1.56347
1.39443	1.3255	1.30171	1.3255	1.39443	1.50231	1.64148
1.5	1.43614	1.41421	1.43614	1.5	1.60078	1.73205

