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)

Институт энергетики, информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа № 2.1**
**по дисциплине дискретная математика**
**тема: Алгоритмы порождения комбинаторных объектов**

**Выполнил: студент группы ПВ-223**
**Игнатьев Артур Олегович**
**Проверил: доцент**
**Рязанов Юрий Дмитриевич**
**старший преподаватель**
**Бондаренко Татьяна Владимировна**

Белгород 2022

Лабораторная работа № 2.1

**Тема:** Алгоритмы порождения комбинаторных объектов

**Цель работы:** изучить основные комбинаторные объекты, алгоритмы их порождения, программно реализовать и оценить временную сложность алгоритмов.

## Задания

1. Реализовать алгоритм порождения подмножеств.

2. Построить график зависимости количества всех подмножеств от мощности множества.

3. Построить графики зависимости времени выполнения алгоритмов п.1 на вашей ЭВМ от мощности множества.

4. Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на вашей ЭВМ.

5. Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.

6. Реализовать алгоритм порождения сочетаний.

7. Построить графики зависимости количества всех сочетаний из n по k от k при n= (5, 7, 9).

8. Реализовать алгоритм порождения перестановок.

9. Построить график зависимости количества всех перестановок от мощности множества.

10. Построить графики зависимости времени выполнения алгоритма п.8 на вашей ЭВМ от мощности множества.

11. Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на вашей ЭВМ.

12. Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.

13. Реализовать алгоритм порождения размещений.

14. Построить графики зависимости количества всех размещений из n по k от k при n= (5, 7, 9).

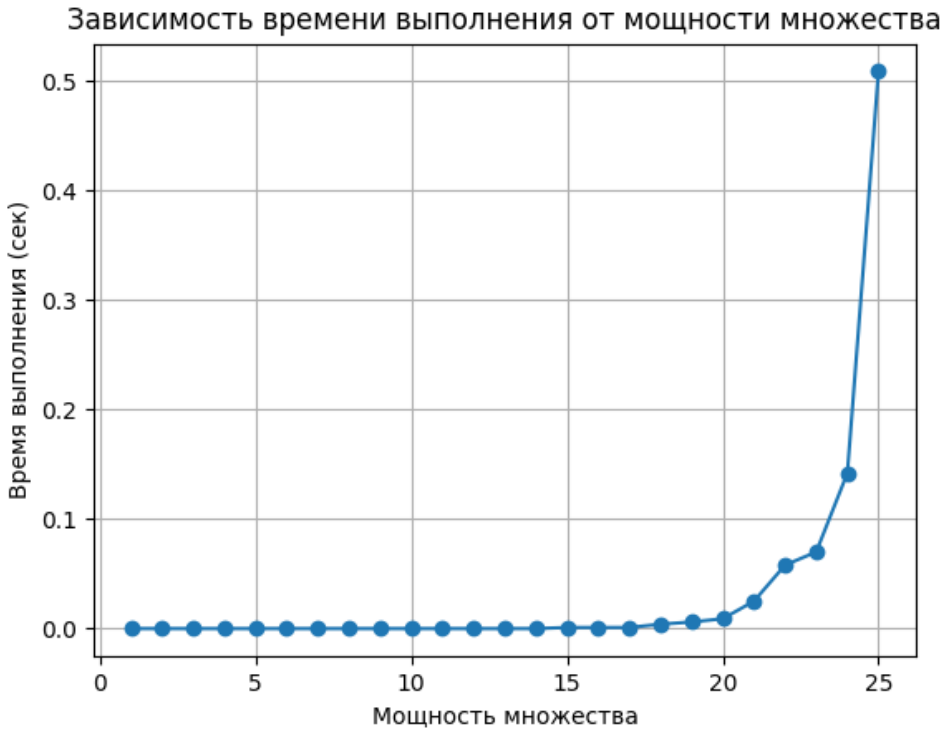
Решение заданий:

1. **Реализовать алгоритм порождения подмножеств.**

```c
#include <stdio.h>

// Функция для печати подмножества
void printSubset(int subset[], int size) {
    printf("{ ");
    for (int i = 0; i < size; i++) {
        printf("%d ", subset[i]);
    }
    printf("}\n");
}

// Рекурсивная функция для генерации всех подмножеств
void generateSubsets(int set[], int subset[], int n, int
subsetSize, int nextIndex) {
    // Печатаем текущее подмножество
    printSubset(subset, subsetSize);

    // Генерируем оставшиеся подмножества с помощью рекурсии
    for (int i = nextIndex; i < n; i++) {
        subset[subsetSize] = set[i];
        generateSubsets(set, subset, n, subsetSize + 1, i +
1);
    }
}

// Функция для вызова генерации подмножеств
void generateAllSubsets(int set[], int n) {
    int subset[n]; // Временный массив для хранения
подмножества

    generateSubsets(set, subset, n, 0, 0);
}

int main() {
    int set[] = {1, 2, 3};
    int n = sizeof(set) / sizeof(set[0]);

    generateAllSubsets(set, n);

    return 0;
}
```

2. **Построить график зависимости количества всех подмножеств от мощности множества.**


**3. Построить графики зависимости времени выполнения алгоритмов п.1 на вашей ЭВМ от мощности множества.**



**4. Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на вашей ЭВМ.**

| Кол-во элем. | Секунды | Секунды по формуле (a*2^кол-во элем.) | | a |
|---|---|---|---|---|
| 15 | 0,001 | 0,00047514 | | 0,0000000145 |
| 16 | 0,001 | 0,00095027 | | |
| 17 | 0,001 | 0,00190054 | | |
| 18 | 0,004 | 0,00380109 | | |
| 19 | 0,006 | 0,00760218 | | |
| 20 | 0,009 | 0,01520435 | | |
| 21 | 0,025 | 0,03040870 | | |
| 22 | 0,058 | 0,06081741 | | |
| 23 | 0,070 | 0,12163482 | | |
| 24 | 0,142 | 0,24326963 | | |
| 25 | 0,509 | 0,48653926 | | |
| 26 | | 0,97307853 | | |
| 27 | | 1,94615706 | | |
| 28 | | 3,89231411 | | |
| 29 | | 7,78462822 | | |
| 30 | | 15,56925645 | | |
| 31 | | 31,13851290 | | |
| 32 | | 62,27702579 | | |
| 33 | | 124,55405158 | | |
| 34 | | 249,10810317 | | |
| 35 | | 498,21620634 | | |
| 36 | | 996,43241267 | | |
| 37 | | 1992,86482534 | час | |
| 38 | | 3985,72965069 | | |
| 39 | | 7971,45930138 | | |
| 40 | | 15942,91860275 | | |
| 41 | | 31885,83720550 | | |
| 42 | | 63771,67441101 | сутки | |
| 43 | | 127543,34882202 | | |
| 44 | | 255086,69764403 | | |
| 45 | | 510173,39528806 | | |
| 46 | | 1020346,79057613 | | |
| 47 | | 2040693,58115226 | месяц | |
| 48 | | 4081387,16230451 | | |
| 49 | | 8162774,32460902 | | |
| 50 | | 16325548,64921800 | год | |

Час-37

Сутки-42

Месяц-47

Год-50

**5. Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.**

| Кол-во элем. | В 10 раз быстрее | | В 100 раз быстрее | |
|---|---|---|---|---|
| 15 | 0,00004751 | | 4,75136E-06 | |
| 16 | 0,00009503 | | 9,50272E-06 | |
| 17 | 0,00019005 | | 1,90054E-05 | |
| 18 | 0,00038011 | | 3,80109E-05 | |
| 19 | 0,00076022 | | 7,60218E-05 | |
| 20 | 0,00152044 | | 0,000152044 | |
| 21 | 0,00304087 | | 0,000304087 | |
| 22 | 0,00608174 | | 0,000608174 | |
| 23 | 0,01216348 | | 0,001216348 | |
| 24 | 0,02432696 | | 0,002432696 | |
| 25 | 0,04865393 | | 0,004865393 | |
| 26 | 0,09730785 | | 0,009730785 | |
| 27 | 0,19461571 | | 0,019461571 | |
| 28 | 0,38923141 | | 0,038923141 | |
| 29 | 0,77846282 | | 0,077846282 | |
| 30 | 1,55692564 | | 0,155692564 | |
| 31 | 3,11385129 | | 0,311385129 | |
| 32 | 6,22770258 | | 0,622770258 | |
| 33 | 12,45540516 | | 1,245540516 | |
| 34 | 24,91081032 | | 2,491081032 | |
| 35 | 49,82162063 | | 4,982162063 | |
| 36 | 99,64324127 | | 9,964324127 | |
| 37 | 199,28648253 | | 19,92864825 | |
| 38 | 398,57296507 | | 39,85729651 | |
| 39 | 797,14593014 | | 79,71459301 | |
| 40 | 1594,29186028 | | 159,429186 | |
| 41 | 3188,58372055 | час | 318,8583721 | |
| 42 | 6377,16744110 | | 637,7167441 | |
| 43 | 12754,33488220 | | 1275,433488 | |
| 44 | 25508,66976440 | | 2550,866976 | час |
| 45 | 51017,33952881 | день | 5101,733953 | |
| 46 | 102034,67905761 | | 10203,46791 | |
| 47 | 204069,35811523 | | 20406,93581 | |
| 48 | 408138,71623045 | | 40813,87162 | |
| 49 | 816277,43246090 | | 81627,74325 | день |
| 50 | 1632554,86492180 | месяц | 163255,4865 | |
| 51 | 3265109,72984361 | | 326510,973 | |
| 52 | 6530219,45968722 | | 653021,946 | |
| 53 | 13060438,91937440 | | 1306043,892 | месяц |
| 54 | 26120877,83874890 | год | 2612087,784 | |
| 55 | 52241755,67749780 | | 5224175,568 | |
| 56 | 104483511,35499600 | | 10448351,14 | |
| 57 | 208967022,70999100 | | 20896702,27 | год |
| 58 | 417934045,41998200 | | 41793404,54 | |

В 10 раз быстрее:

Час-41

День-45

Месяц-50

Год-54

В 100 раз быстрее:

Час-44

День-49

Месяц-53

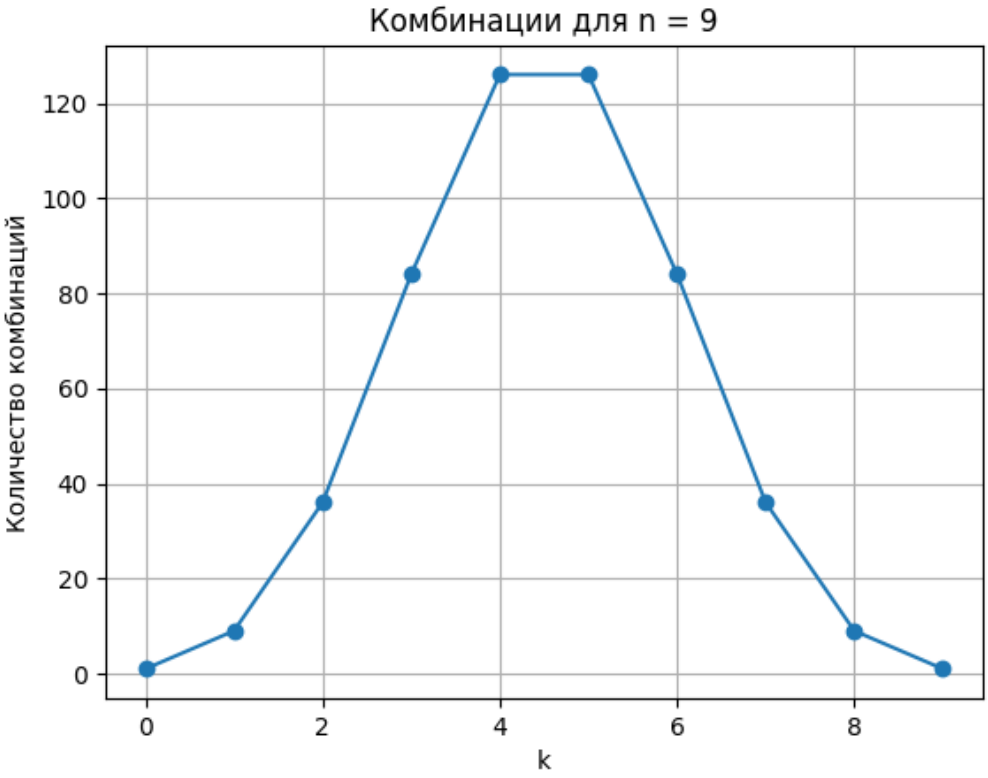
Год-57

**6. Реализовать алгоритм порождения сочетаний.**

```c
#include <stdio.h>

void generate_combinations(int arr[], int data[], int start,
                           int end, int index, int r) {
    if (index == r) {
        // Печатаем текущую комбинацию
        for (int i = 0; i < r; i++) {
            printf("%d ", data[i]);
        }
        printf("\n");
        return;
    }

    // Генерируем комбинации, начиная с элемента start
    for (int i = start; i <= end && end - i + 1 >= r - index;
i++) {
        data[index] = arr[i];
        generate_combinations(arr, data, i + 1, end, index + 1,
r);
    }
}

void print_combinations(int arr[], int n, int r) {
    int data[r];
    generate_combinations(arr, data, 0, n - 1, 0, r);
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr) / sizeof(arr[0]);
    int r = 3;

    print_combinations(arr, n, r);

    return 0;
}
```

**7. Построить графики зависимости количества всех сочетаний из n по k от k при n= (5, 7, 9).**

Для построения графиков была использована формула сочетания C(n, k) = n! / (k! * (n - k)!), где n - количество элементов, а k - размер сочетания.



Комбинации для n = 9
Количество комбинаций
k
0
20
40
60
80
100
120
0
2
4
6
8

**8. Реализовать алгоритм порождения перестановок.**

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void generatePermutations(int elements[], int size, int index) {
    // Базовый случай: если достигли конца массива, выводим
перестановку
    if (index == size - 1) {
        for (int i = 0; i < size; i++) {
            printf("%d ", elements[i]);
        }
        printf("\n");
        return;
    }

    // Генерируем перестановки для оставшейся части массива
    for (int i = index; i < size; i++) {
        // Меняем текущий элемент с элементом на позиции index
        swap(&elements[index], &elements[i]);

        // Рекурсивно генерируем перестановки для оставшейся
части
        generatePermutations(elements, size, index + 1);

        // Восстанавливаем исходный порядок элементов перед
следующей итерацией
        swap(&elements[index], &elements[i]);
    }
}

int main() {
    int elements[] = {1, 2, 3};
    int size = sizeof(elements) / sizeof(elements[0]);

    generatePermutations(elements, size, 0);

    return 0;
}
```

**9. Построить график зависимости количества всех перестановок от мощности множества.**

**10. Построить графики зависимости времени выполнения алгоритма п.8 на вашей ЭВМ от мощности множества.**

Модифицированная программа:

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

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void generatePermutations(int elements[], int size, int index) {
    // Базовый случай: если достигли конца массива, выводим
перестановку
    if (index == size - 1) {
        for (int i = 0; i < size; i++) {
            printf("%d ", elements[i]);
        }
        printf("\n");
        return;
    }

    // Генерируем перестановки для оставшейся части массива
    for (int i = index; i < size; i++) {
        // Меняем текущий элемент с элементом на позиции index
        swap(&elements[index], &elements[i]);

        // Рекурсивно генерируем перестановки для оставшейся
части
        generatePermutations(elements, size, index + 1);

        // Восстанавливаем исходный порядок элементов перед
следующей итерацией
        swap(&elements[index], &elements[i]);
    }
}

int main() {
    SetConsoleOutputCP(CP_UTF8);

    int elements[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int size = sizeof(elements) / sizeof(elements[0]);

    // Измерение времени выполнения
    clock_t start = clock();

    generatePermutations(elements, size, 0);
```

```
    clock_t end = clock();
    double time_spent = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Время выполнения: %f сек\n", time_spent);

    return 0;
}
```

**11. Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на вашей ЭВМ.**

| Мощность множ. | Время, с | Кол-во перестановок | |
|---|---|---|---|
| 1 | 0.001 | 1 | |
| 2 | 0.001 | 2 | |
| 3 | 0.004 | 6 | |
| 4 | 0.021 | 24 | |
| 5 | 0.144 | 120 | |
| 6 | 0.999 | 720 | |
| 7 | 7.228 | 5040 | |
| 8 | 56.568 | 40320 | |
| 9 | 557.964 | 362880 | час |
| 10 | 6409.956 | 3628800 | |
| 11 | 65437.592 | 39916800 | день |
| 12 | 720812.741 | 479001600 | месяц |
| 13 | 7928930.505 | 4790016000 | год |
| 14 | 87218222.557 | 62270208000 | |
| 15 | 961400472.02 | 87178291200 | |

Час-9

Сутки-11

Месяц-12

Год-13

**12. Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.**

| Можность множ. | В 10 раз быстрее | Кол-во перестановок | В 100 раз быстрее | | |
|---|---|---|---|---|---|
| 1 | 0,0001 | 1 | 0,00001 | | |
| 2 | 0,0001 | 2 | 0,00001 | | |
| 3 | 0,0001 | 6 | 0,00001 | | |
| 4 | 0,0004 | 24 | 0,00004 | | |
| 5 | 0,0144 | 120 | 0,00144 | | |
| 6 | 0,0999 | 720 | 0,00999 | | |
| 7 | 0,7228 | 5040 | 0,07228 | | |
| 8 | 5,6568 | 40320 | 0,56568 | | |
| 9 | 55,7964 | 362880 | 5,57964 | | |
| 10 | 640,9956 | 3628800 | 64,09956 | час | |
| 11 | 6543,7592 | 39916800 | 654,37592 | | час |
| 12 | 72081,2741 | 479001600 | 7208,12741 | сутки | |
| 13 | 792893,0505 | 4790016000 | 79289,30505 | месяц | сутки |
| 14 | 8721822,2557 | 62270208000 | 872182,22557 | год | месяц |
| 15 | 96140047,2020 | 87178291200 | 9614004,72020 | | год |

В 10 раз быстрее:

Час-10

Сутки-12

Месяц-13

Год-14

В 100 раз быстрее:

Час-11

Сутки-13

Месяц-14

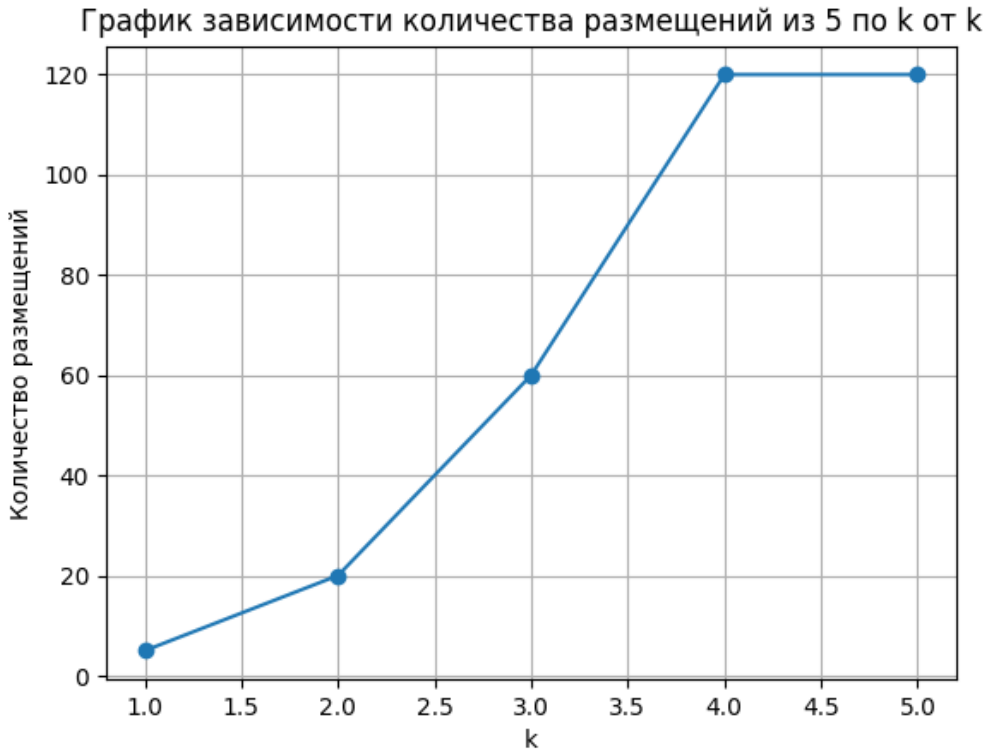
Год-15

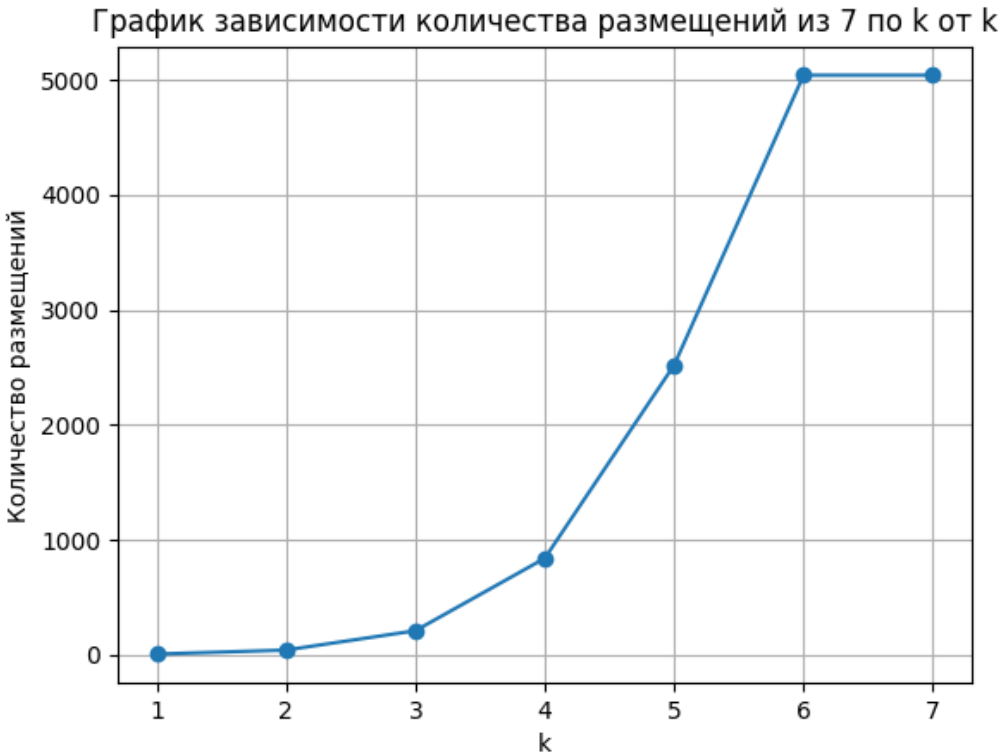
**13. Реализовать алгоритм порождения размещений.**

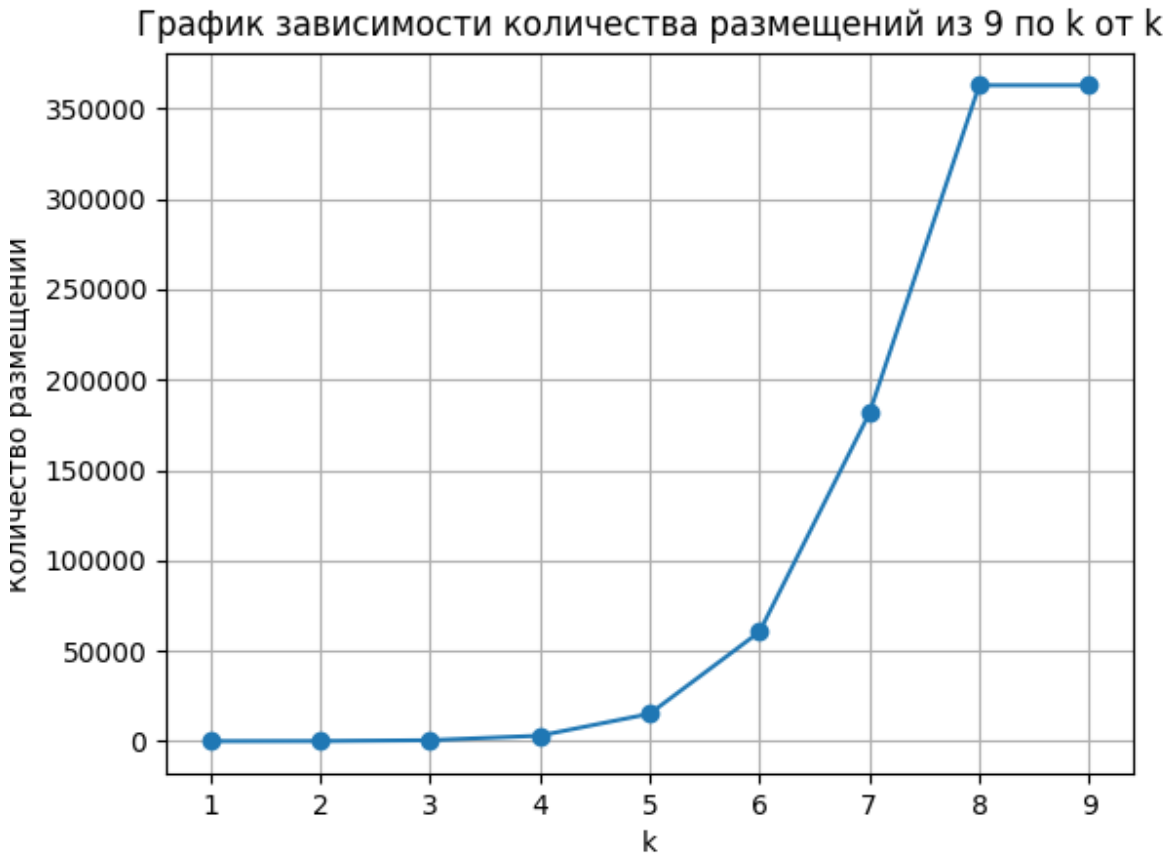
```c
#include <stdio.h>
#include <windows.h>

// Функция для обмена двух элементов
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Функция для печати размещения
void printArr(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// Рекурсивная функция для генерации размещений
void generatePermutations(int arr[], int size, int n) {
    if (size == 1) {
        printArr(arr, n);
        return;
    }

    for (int i = 0; i < size; i++) {
        generatePermutations(arr, size - 1, n);

        // Если размер массива нечетный, меняем первый элемент с
последним
        if (size % 2 == 1) {
            swap(&arr[0], &arr[size - 1]);
        }
            // Если размер массива четный, меняем i-й элемент с
последним
        else {
            swap(&arr[i], &arr[size - 1]);
        }
    }
}

int main() {
    SetConsoleOutputCP(CP_UTF8);

    int n;
    printf("Введите размер массива: ");
    scanf("%d", &n);

    int arr[n];
    printf("Введите элементы массива: ");
    for (int i = 0; i < n; i++) {
```

```
        scanf("%d", &arr[i]);
    }

    printf("Размещения:\n");
    generatePermutations(arr, n, n);

    return 0;
}
```

**14. Построить графики зависимости количества всех размещений из n по k от k при n= (5, 7, 9).**






Вывод: на этой лабораторной работе я изучил основные комбинаторные объекты, алгоритмы их порождения, программно реализовал и оценил временную сложность алгоритмов.