Учреждение образования “БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ”

Лабораторная работа №2

«Комбинаторные алгоритмы решения оптимизационных задач»

Выполнила:

студентка 2 курса 7 группы

Колядко Яна Дмитриевна

Проверил:

Барковский Евгений Валерьевич

ПроТТеоре

Цель работы: приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научиться применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой.

Задание 1

Разработать генератор подмножеств заданного множества.

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " - Генератор множества всех подмножеств -";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация всех подмножеств ";

combi1::subset s1(sizeof(AA) / 2); // создание генератора

int n = s1.getfirst(); // первое (пустое) подмножество

while (n >= 0) // пока есть подмножества

{

std::cout << std::endl << "{ ";

for (int i = 0; i < n; i++)

std::cout << AA[s1.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s1.getnext(); // cледующее подмножество

};

std::cout << std::endl << "всего: " << s1.count() << std::endl;

system("pause");

return 0;

Изображение выглядит как текст

Автоматически созданное описание

Задание 2

Разработать генератор сочетаний.

char AA[][2] = { "A", "B", "C", "D", "E" };

std::cout << std::endl << " --- Генератор сочетаний ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация сочетаний ";

combi2::xcombination xc(sizeof(AA) / 2, 3);

std::cout << "из " << xc.n << " по " << xc.m;

int n = xc.getfirst();

while (n >= 0)

{

std::cout << std::endl << xc.nc << ": { ";

for (int i = 0; i < n; i++)

std::cout << AA[xc.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = xc.getnext();

};

std::cout << std::endl << "всего: " << xc.count() << std::endl;

system("pause");

return 0;

Изображение выглядит как текст

Автоматически созданное описание

Задание 3

Разработать генератор перестановок.

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор перестановок ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация перестановок ";

combi3::permutation p(sizeof(AA) / 2);

\_\_int64 n = p.getfirst();

while (n >= 0)

{

std::cout << std::endl << std::setw(4) << p.np << ": { ";

for (int i = 0; i < p.n; i++)

std::cout << AA[p.ntx(i)] << ((i < p.n - 1) ? ", " : " ");

std::cout << "}";

n = p.getnext();

};

std::cout << std::endl << "всего: " << p.count() << std::endl;

system("pause");

return 0;

Изображение выглядит как текст

Автоматически созданное описание

Задание 4

Разработать генератор размещений.

char AA[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор размещений ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < N; i++)

std::cout << AA[i] << ((i < N - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация размещений из " << N << " по " << M;

combi4::accomodation s(N, M);

int n = s.getfirst();

while (n >= 0) {

std::cout << std::endl << std::setw(2) << s.na << ": { ";

for (int i = 0; i < 3; i++)

std::cout << AA[s.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s.getnext();

};

std::cout << std::endl << "всего: " << s.count() << std::endl;

system("pause");

return 0;

Изображение выглядит как текст

Автоматически созданное описание

Задание 5

Решить в соответствии с вариантом задачу: (Вариант 2, 6, 10, 14) упрощенную о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.).

clock\_t t1 = 0, t2 = 0;

t1 = clock();

int V = 300, // вместимость рюкзака

v[NN] = { }, // размер предмета каждого типа

c[NN] = { }; // стоимость предмета каждого типа

for (int i = 0; i < NN; i++)

{

v[i] = 10 + rand() % 300;

c[i] = 5 + rand() % 55;

}

short m[NN]; // количество предметов каждого типа {0,1}

int maxcc = knapsack\_s(

V, // [in] вместимость рюкзака

NN, // [in] количество типов предметов

v, // [in] размер предмета каждого типа

c, // [in] стоимость предмета каждого типа

m // [out] количество предметов каждого типа

);

std::cout << std::endl << "-------- Задача о рюкзаке ---------";

std::cout << std::endl << "- количество предметов : " << NN;

std::cout << std::endl << "- вместимость рюкзака : " << V;

std::cout << std::endl << "- размеры предметов : ";

for (int i = 0; i < NN; i++) std::cout << v[i] << " ";

std::cout << std::endl << "- стоимости предмета : ";

for (int i = 0; i < NN; i++) std::cout << c[i] << " ";

std::cout << std::endl << "- стоимости предметов : ";

for (int i = 0; i < NN; i++) std::cout << v[i] \* c[i] << " ";

std::cout << std::endl << "- оптимальная стоимость рюкзака: " << maxcc;

std::cout << std::endl << "- вес рюкзака: ";

int s = 0; for (int i = 0; i < NN; i++) s += m[i] \* v[i];

std::cout << s;

std::cout << std::endl << "- выбраны предметы: ";

for (int i = 0; i < NN; i++) std::cout << " " << m[i];

std::cout << std::endl << std::endl;

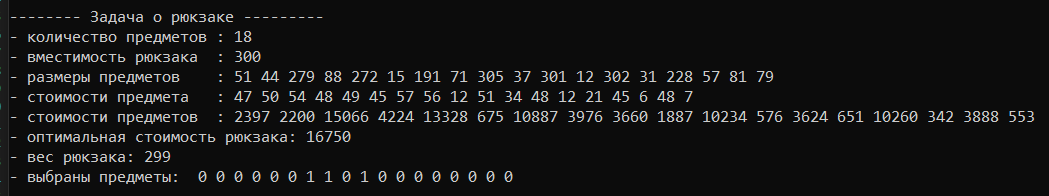
t2 = clock();

std::cout << std::endl << "продолжительность (у.е): " << (t2 - t1);

std::cout << std::endl;

system("pause");

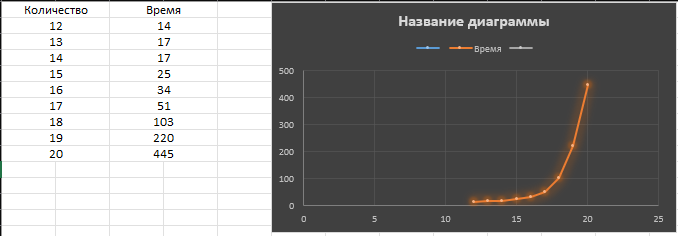
return 0;



Задание 6

Исследовать зависимость времени вычисления необходимое для решения задачи (в соответствии с вариантом) от размерности задачи:

(Вариант 2, 6, 10, 14) упрощенную о рюкзаке (количество предметов 12–20 шт.).



Вывод: в данной лабораторной работе были приобретены навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научились применять разработанные генераторы для решения задачи о рюкзаке (упрощенную).

**Алгоритм генерации перестановки** – Наиболее известным методом построения множества  всех перестановок конечного множества  является ***алгоритм Джонсона – Троттера***.

В алгоритме используется понятие ***мобильного элемента***. Элемент  последовательности элементов множества  называется мобильным, если соответствующая ему стрелка указывает на меньший соседний элемент.

Построение множества всех перестановок с помощью алгоритма Джонсона – Троттера сводится к следующей процедуре:

1. Построить первую перестановку. Первая перестановка – это последовательность всех элементов множества  перечисленных в порядке возрастания. Стрелки всех элементов последовательности направлены влево.
2. Найти наибольший мобильный элемент в текущей перестановке. Если в последовательности нет мобильного элемента, то построены все перестановки элементов множества  – алгоритм закончил свою работу.

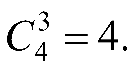
3. Поменять местами наибольший мобильный элемент и элемент, на который указывает стрелка наибольшего мобильного элемента.

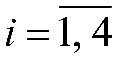
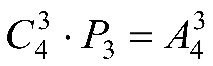
4. Найти все элементы, большие, чем мобильный элемент (если они есть) и изменить их стрелки на противоположное направление.

5. Перейти к пункту 2.

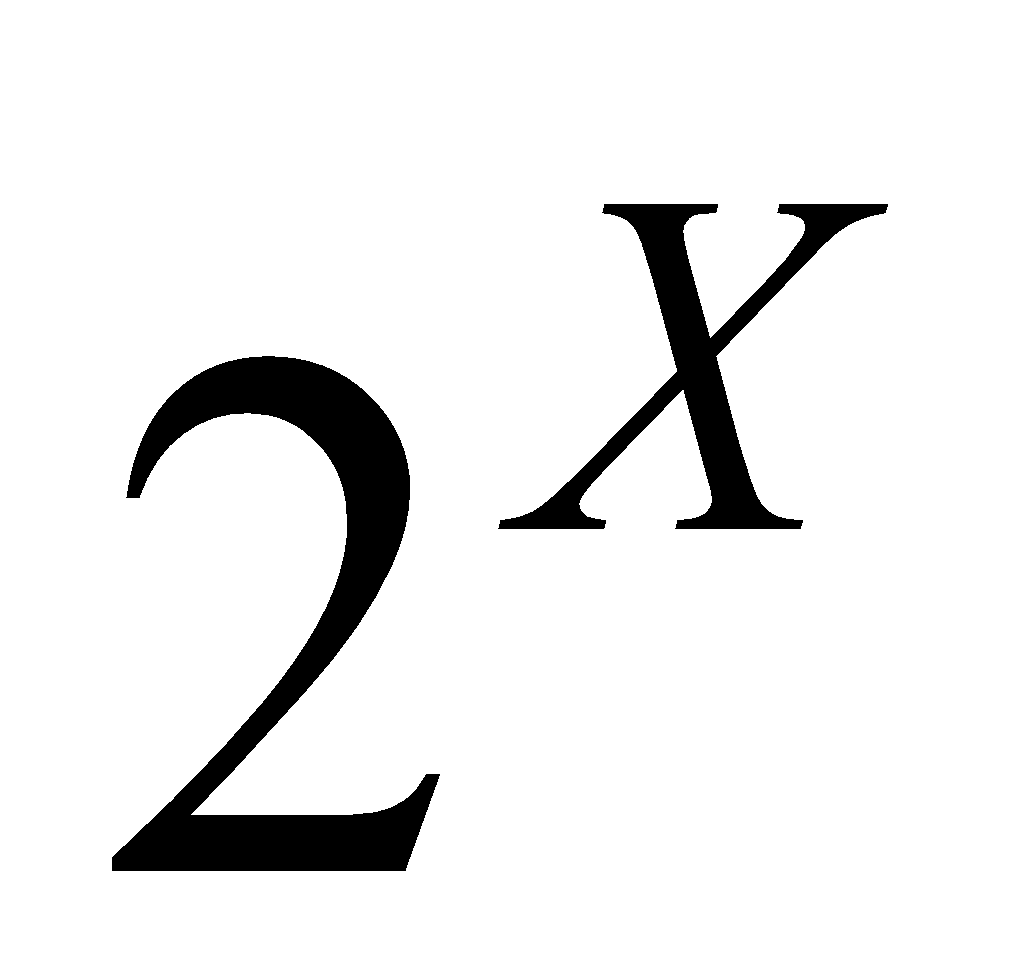
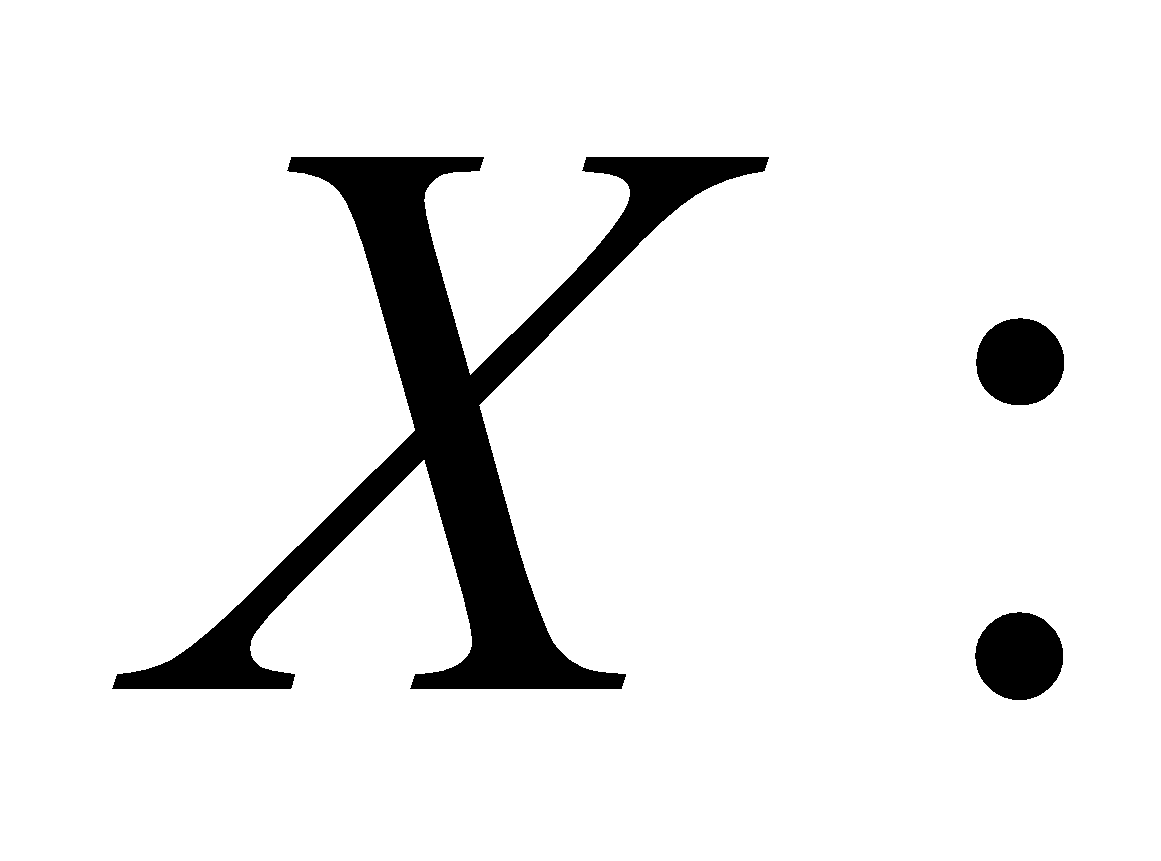
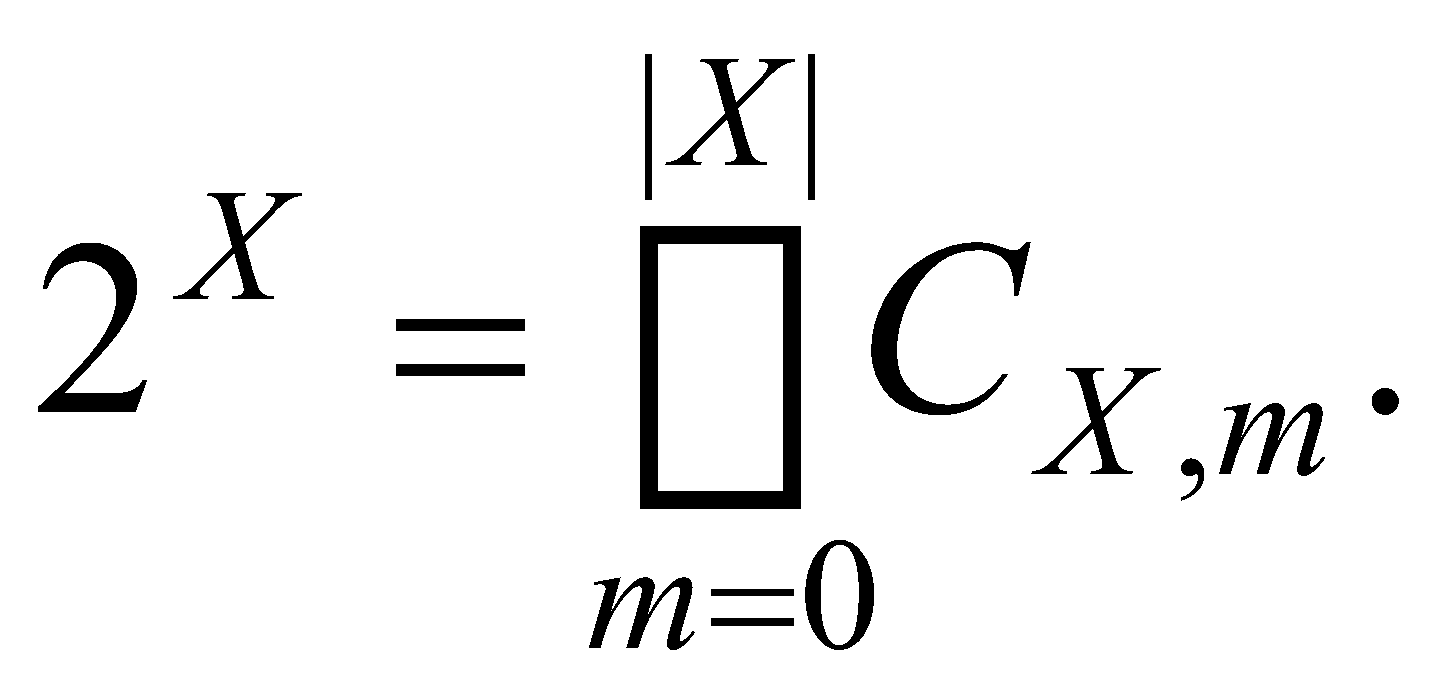
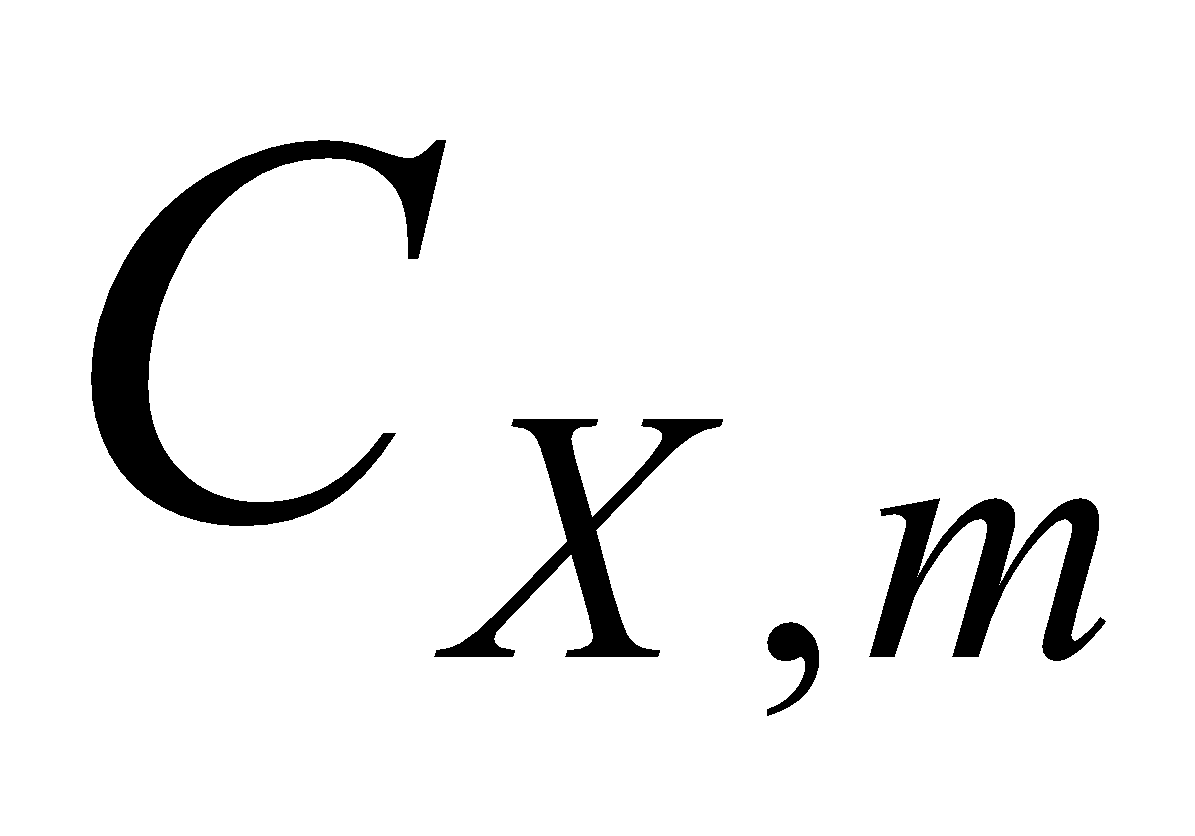
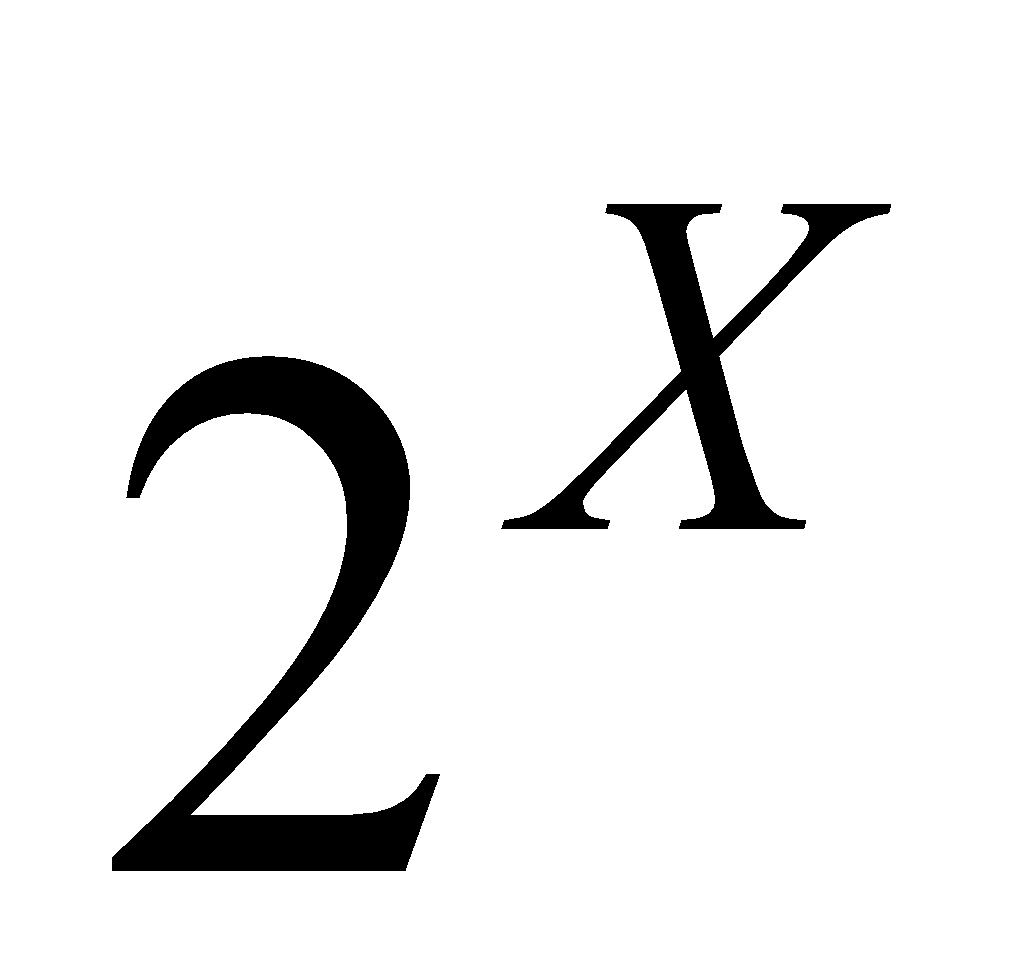
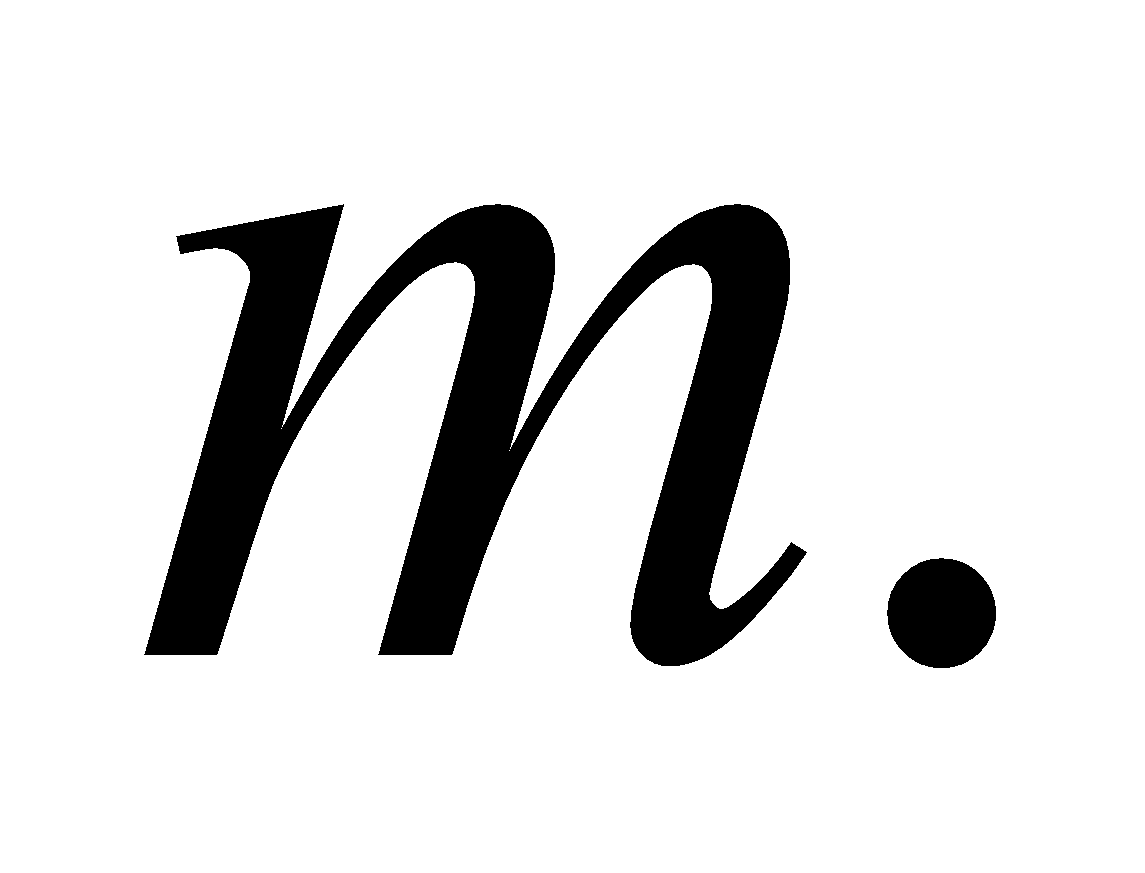
**Алгоритм генерации размещения**

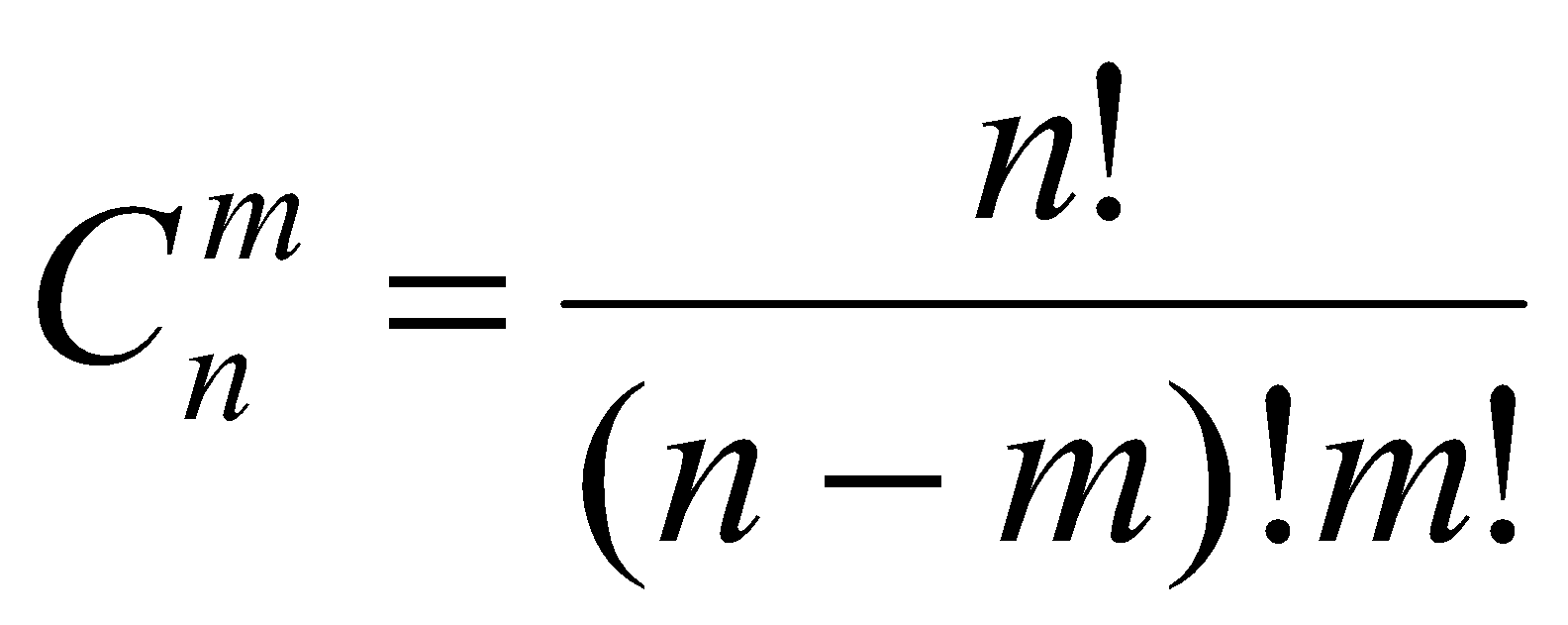
Генерация размещений осуществляется в три этапа:

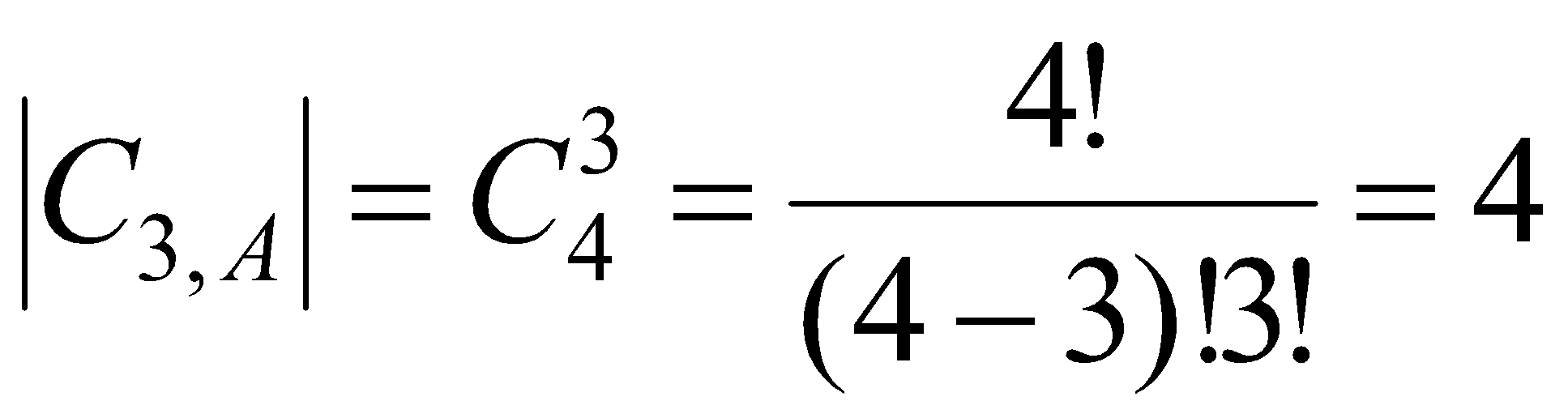
1. Для множества  формируется множество всех сочетаний по три элемента. Таких сочетаний будет 

2. Для элементов каждого сочетания ,  генерируется все перестановки  Каждому сочетанию будет соответствовать  перестановок. Таким образом, всего будет построено  перестановок, которые в совокупности представляют собой все размещения множества 

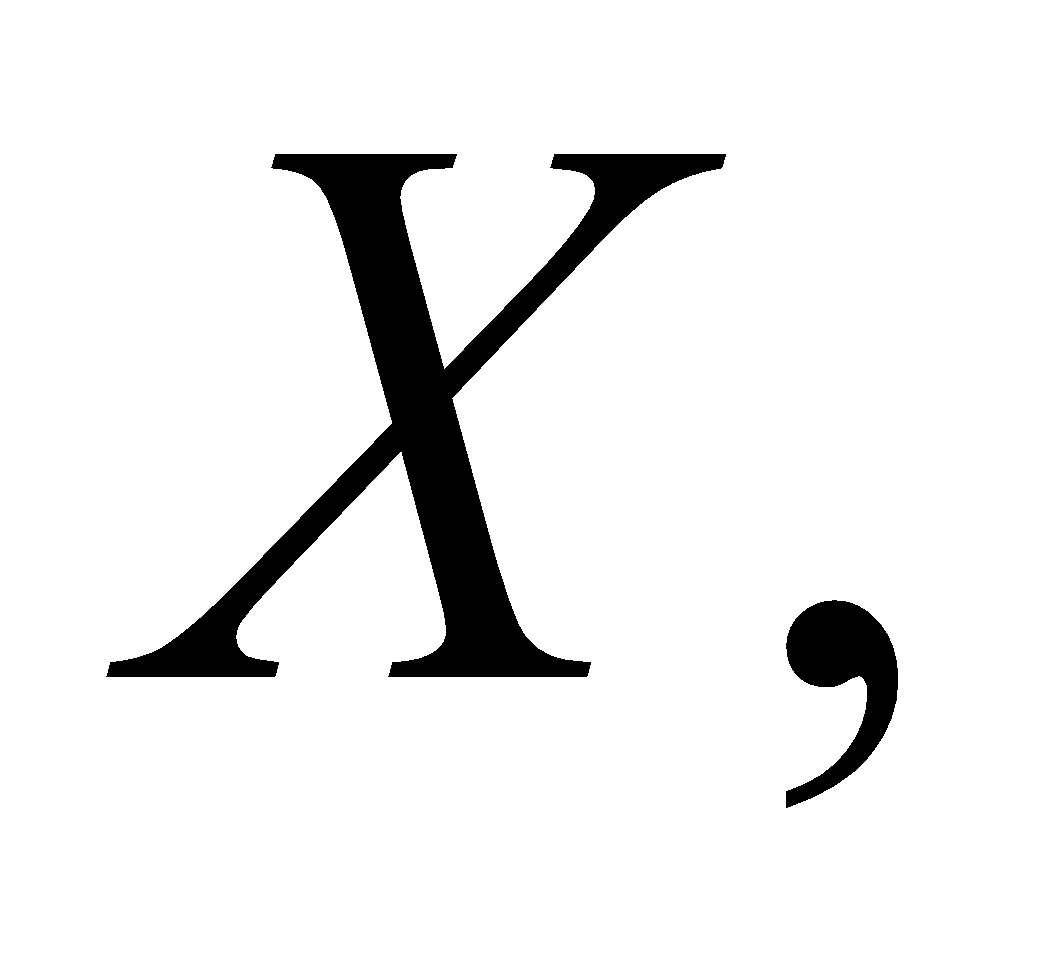
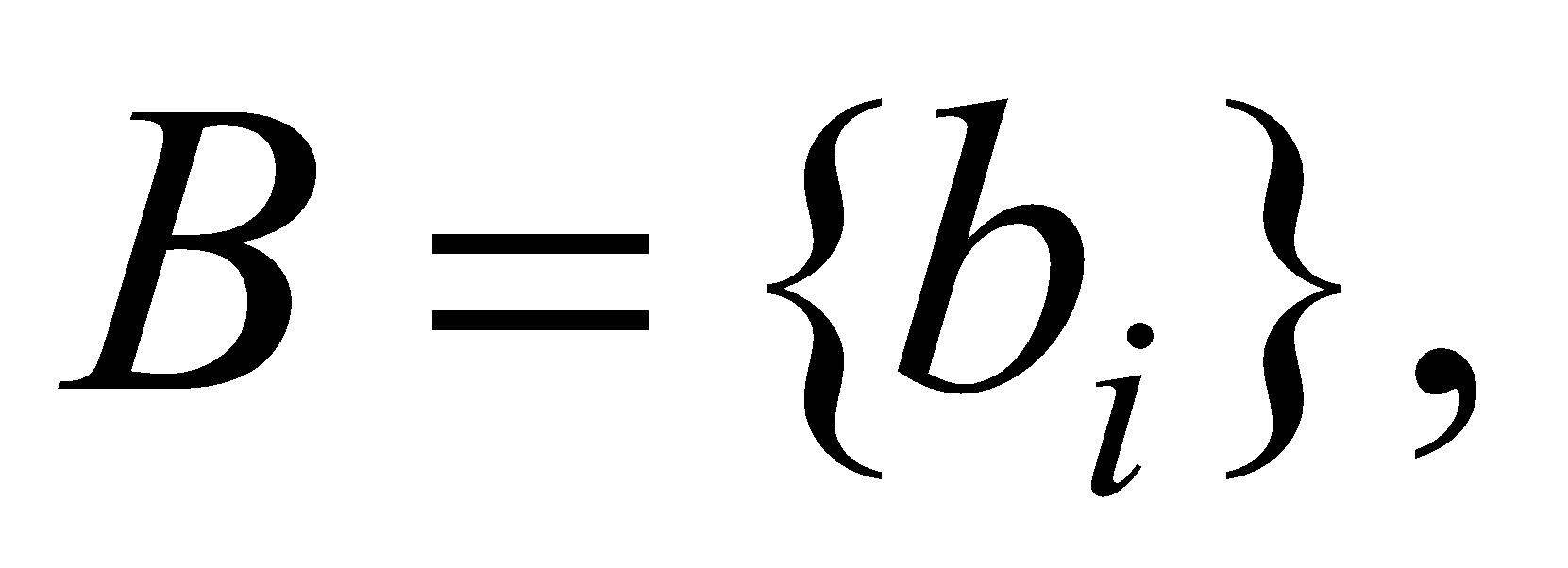
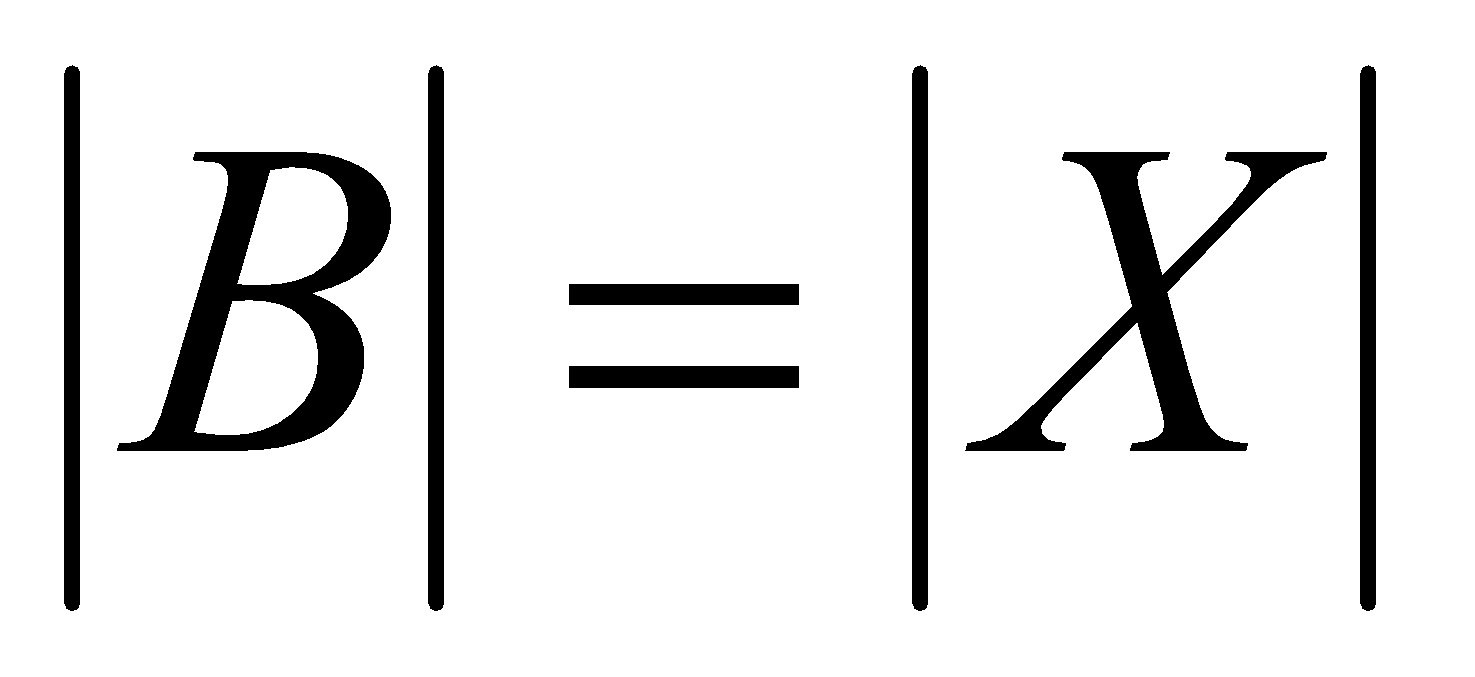
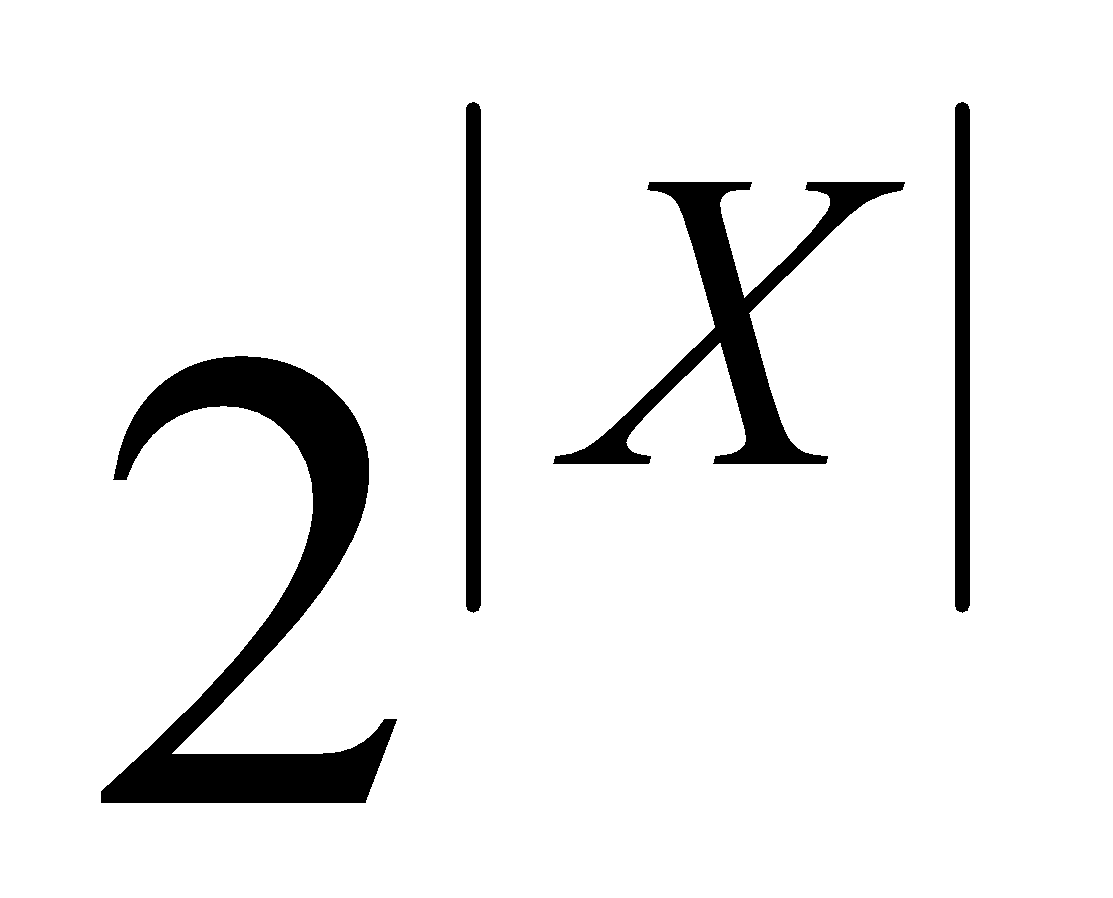
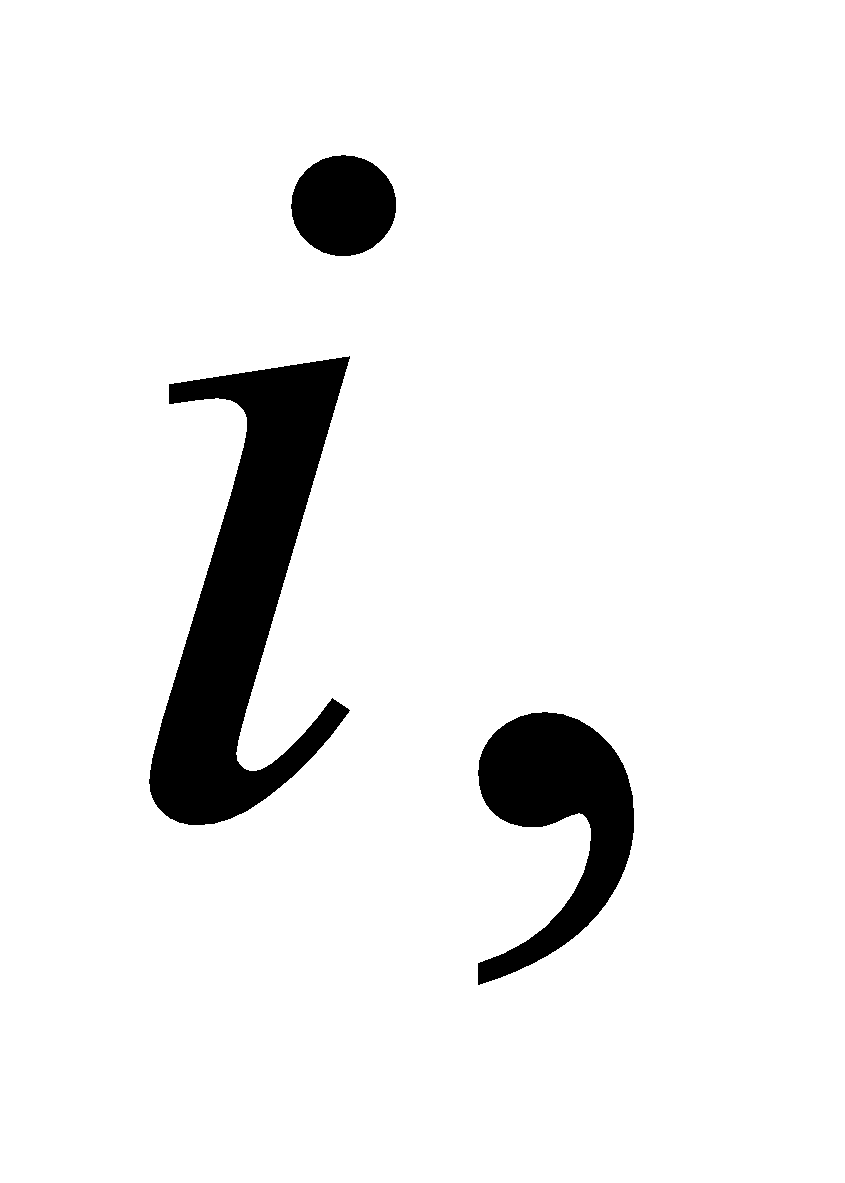
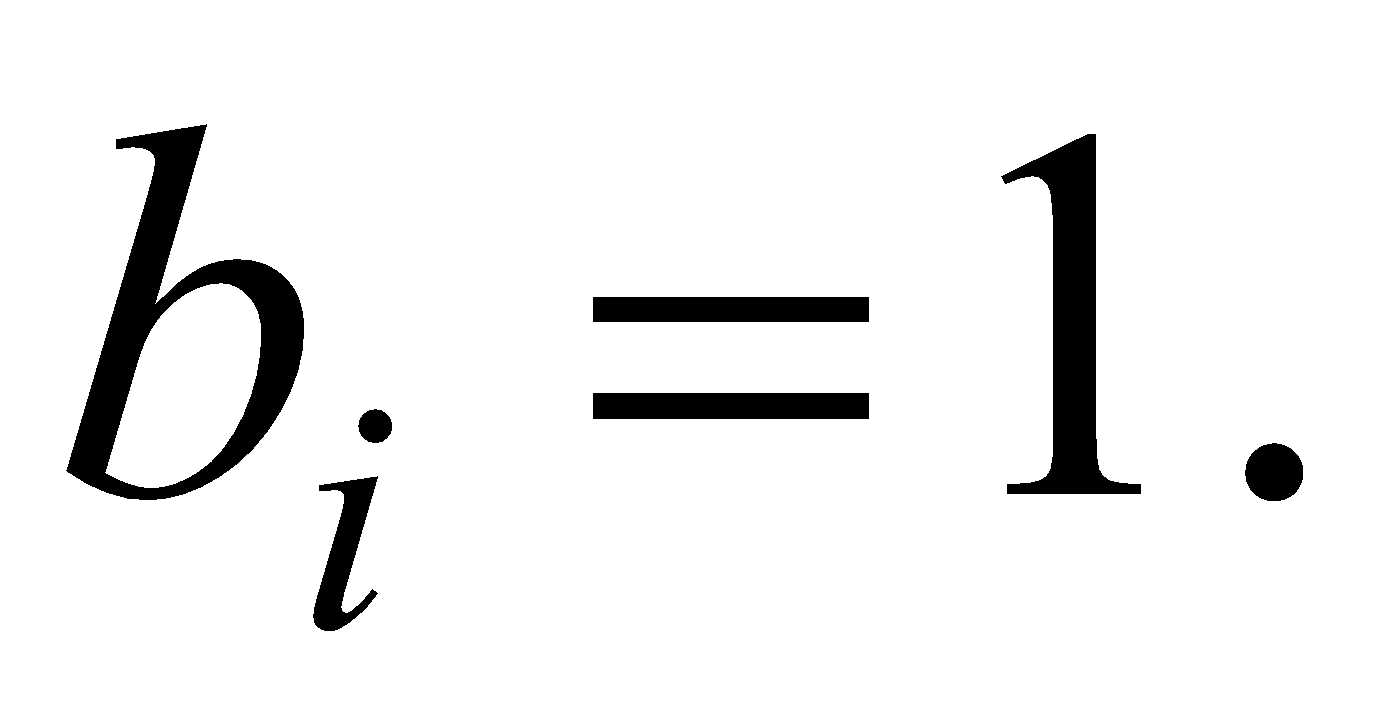
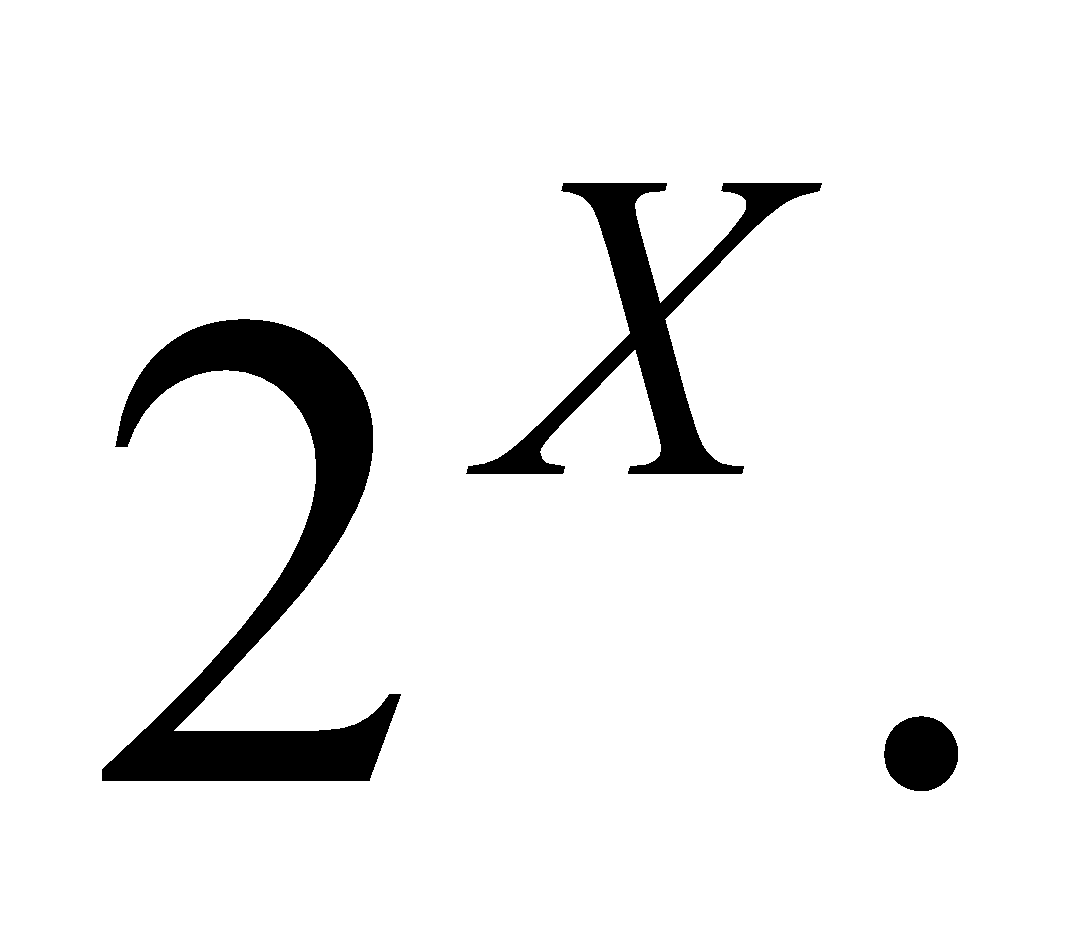
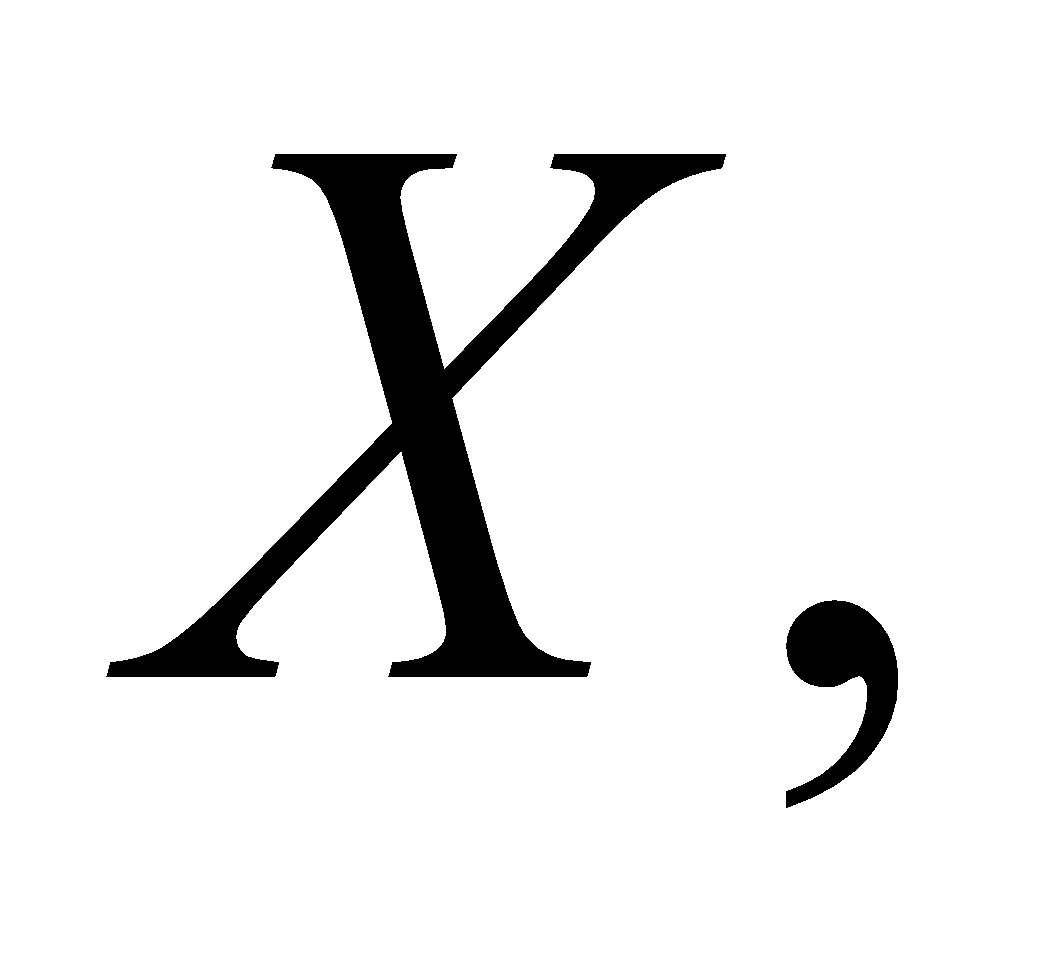
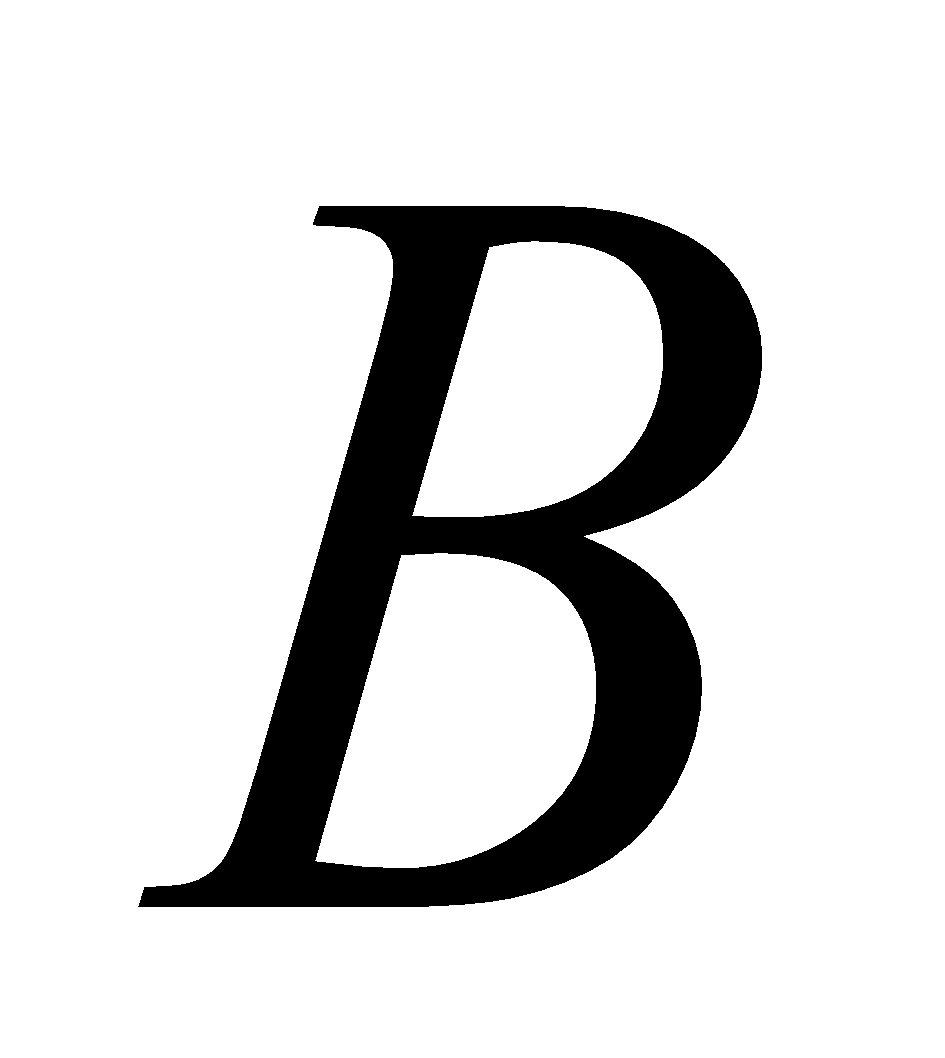
3. Применяя полученные на предыдущем шаге размещения в качестве индексов для элементов множества  формируется множество  всех размещений по 3.

**Алгоритм генерации сочетания** – Булеан  можно рассматривать как объединение всевозможных сочетаний, построенных из элементов множества   Поэтому генерация множества  может быть сведена к генерации булеана и выбору из него всех подмножеств с мощностью 

****

** **

**Алгоритм генерации множества всех подмножеств**

1. Пронумеровать элементы заданного множества  начиная с нуля.
2. Сформировать битовую последовательность  состоящую из  двоичных нулей. Пронумеровать элементы этой последовательности справа налево, начиная с нуля.
3. Последовательно выполнить шаги 4 и 5 алгоритма  раз.
4. Выбрать из множества  элементы с номерами  для которых  Полученное подмножество будет являться элементом булеана  В первом случае, не будет выбран ни один элемент множества  так как исходная последовательность  состоит только из нулей.
5. Интерпретируя битовую последовательность как целое положительное число, увеличить это число на единицу.

Группы элементов, состоящие из одних и тех же элементов и отличающиеся друг от друга только их порядком, называются перестановками этих элементов.

Размещением (из n по k) называется упорядоченный набор из k различных элементов из некоторого множества различных n элементов.

Сочетанием из n по k называется набор из k элементов, выбранных из n-элементного множества, в котором не учитывается порядок элементов.