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

**Лабораторная работа №2.1**

по дисциплине: Дискретная математика

тема: «Алгоритмы порождения комбинаторных объектов»

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверили:

Рязанов Юрий Дмитриевич

Белгород 2022 г.

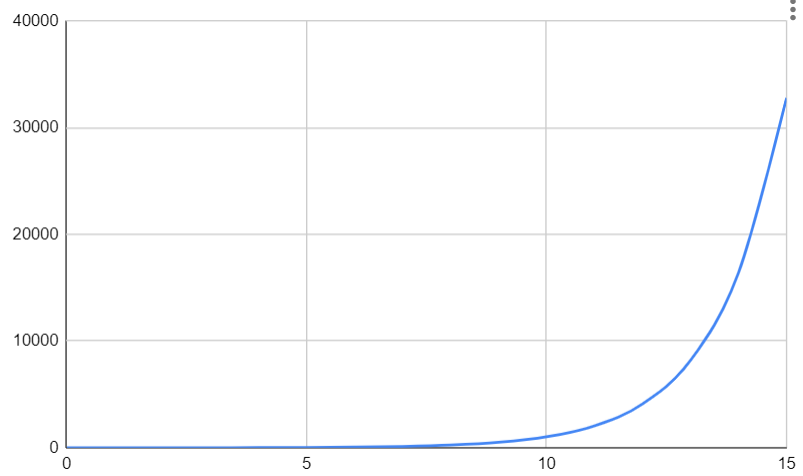
**Цель занятия:** изучить основные комбинаторные объекты, алгоритмы их порождения, программно реализовать и оценить временную сложность алгоритмов.

**Задания**

1. Реализовать алгоритм порождения подмножеств.

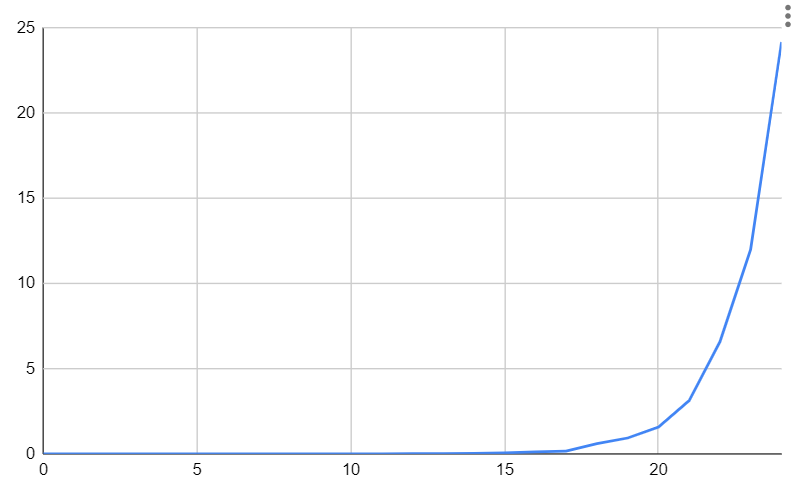
#include **<iostream>**#include **"vector"**#define **TIME\_TEST**(testCode, time) { \  
 clock\_t start\_time = clock(); \  
 testCode \  
clock\_t end\_time = clock();\  
clock\_t sort\_time = end\_time - start\_time; \  
time = (**double**) sort\_time / **CLOCKS\_PER\_SEC**;\  
}  
  
**using namespace** std;  
  
**void** \_getSubArray(**int** i, **const** vector<**int**> &set, vector<**int**> arrayCur,  
 vector<vector<**int**>> &subArrays) {  
 **for** (**int** x = 0; x <= 1; x++) {  
 **if** (i == set.size()) {  
 subArrays.push\_back(arrayCur);  
 **return**;  
 } **else** {  
 **if** (x)  
 arrayCur.push\_back(set[i]);  
 \_getSubArray(i + 1, set, arrayCur, subArrays);  
 }  
 }  
}  
  
**void** getSubArray(**const** vector<**int**> &set, vector<vector<**int**>> &subArrays) {  
 vector<**int**> arrayCur;  
 \_getSubArray(0, set, arrayCur, subArrays);  
}  
  
  
**int** main() {  
 **int** maxSize = 5;  
  
 vector<**int**> set(maxSize);  
 **for** (**int** i = 0; i < maxSize; i++) {  
 set[i] = i + 1;  
 }  
  
 vector<vector<**int**>> subArrays;  
 getSubArray(set, subArrays);  
  
 **for** (**auto** v : subArrays) {  
 **for** (**auto** v1 : v)  
 cout << v1 << **" "**;  
 cout << **"\n"**;  
 }  
  
}

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.

#include **<iostream>**#include **"vector"**#define **TIME\_TEST**(testCode, time) { \  
 clock\_t start\_time = clock(); \  
 testCode \  
clock\_t end\_time = clock();\  
clock\_t sort\_time = end\_time - start\_time; \  
time = (**double**) sort\_time / **CLOCKS\_PER\_SEC**;\  
}  
  
**using namespace** std;  
  
**void** \_getSubArray(**int** i, **const** vector<**int**> &set, vector<**int**> arrayCur,  
 vector<vector<**int**>> &subArrays) {  
 **for** (**int** x = 0; x <= 1; x++) {  
 **if** (i == set.size()) {  
 subArrays.push\_back(arrayCur);  
 **return**;  
 } **else** {  
 **if** (x)  
 arrayCur.push\_back(set[i]);  
 \_getSubArray(i + 1, set, arrayCur, subArrays);  
 }  
 }  
}  
  
**void** getSubArray(**const** vector<**int**> &set, vector<vector<**int**>> &subArrays) {  
 vector<**int**> arrayCur;  
 \_getSubArray(0, set, arrayCur, subArrays);  
}  
  
  
**int** main() {  
 **int** experimentSize = 20;  
  
 vector<**double**> time(experimentSize);  
 **for** (**int** maxSize = 1; maxSize <= experimentSize; maxSize++) {  
 **TIME\_TEST**({  
 vector<**int**> set(maxSize);  
 **for** (**int** i = 0; i < maxSize; i++) {  
 set[i] = i + 1;  
 }  
  
 vector<vector<**int**>> subArrays;  
 getSubArray(set, subArrays);  
 }, time[maxSize])  
 }  
  
 **for** (**int** i = 0; i < experimentSize; i++) {  
 cout << i << **" "** << time[i] << **"\n"**;  
 }  
}



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

220 / 1.54 = 1 048 576 / 1.54 = 680 893 - Множеств генерируется за одну секунду

2n /680 893 = 3 600 → log2 3600 \* 680 893 = n → **31 за час.**

log23600 \* 24 \* 680 893 = **35 за сутки.**

log23600 \* 24 \* 30 \* 680 893= **40 за месяц.**

log23600 \* 24 \* 365 \* 680 893= **44 за год.**

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.

**В 10 раз быстрее:**

2n / 6 808 930 = 3600  log2 3600 \* 6 808 930 = **34 за час.**

log23600 \* 24 \* 6 808 930 = **39 за сутки.**

log23600 \* 24 \* 30 \* 6 808 930 = **44 за месяц.**

log23600 \* 24 \* 365 \* 6 808 930 = **47 за год.**

**В 100 раз быстрее:**

2n / 68 089 030 = 3600  log2 3600 \* 68 089 300 = **37 за час.**

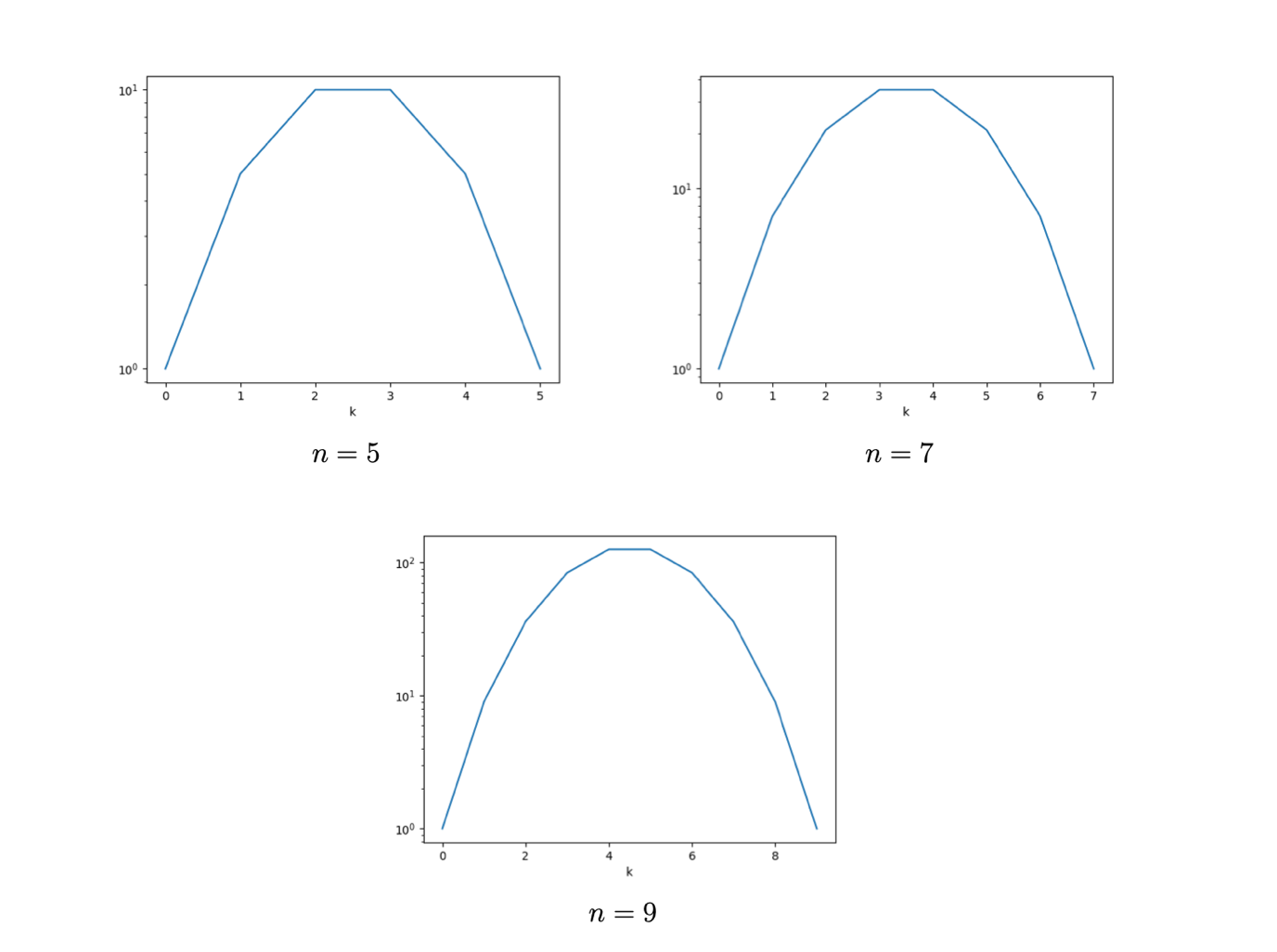
log23600 \* 24 \* 68 089 300 = **42 за сутки.**

log23600 \* 24 \* 30 \* 68 089 300 = **47 за месяц.**

log23600 \* 24 \* 365 \* 68 089 300 = **50 за год.**

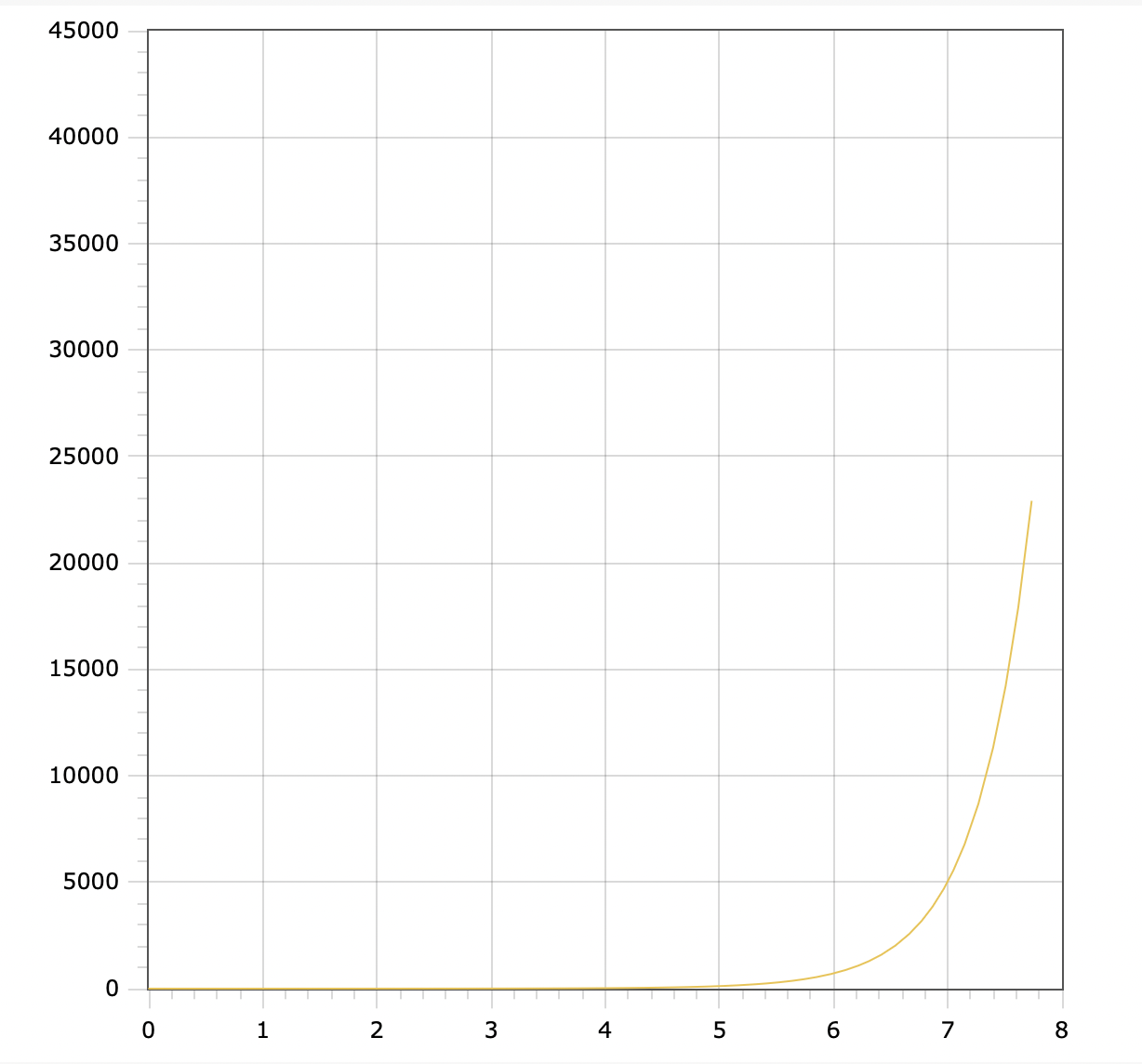
6. Реализовать алгоритм порождения сочетаний.

#include **<iostream>**#include **"vector"  
  
using namespace** std;  
  
**void** \_generateCombinations(**int** n, **int** k, **int** i, **int** b,  
 vector<**int**> inputSet,  
 vector<**int**> generatingSet,  
 vector<vector<**int**>> &combinations) {  
 **for** (**int** x = b; x <= n - k + i; x++) {  
 vector<**int**> copyGeneratingSet = generatingSet;  
 copyGeneratingSet.push\_back(inputSet[x]);  
  
 **if** (i == k) {  
 combinations.push\_back(generatingSet);  
 **return**;  
 } **else** \_generateCombinations(n, k, i + 1, x + 1, inputSet,  
 copyGeneratingSet, combinations);  
 }  
}  
  
**void** generateCombinations(**int** n, **int** k, vector<**int**> inputSet,  
 vector<vector<**int**>>  
 &combinations) {  
 vector<**int**> generatingSet;  
 \_generateCombinations(n, k, 0, 0, inputSet, generatingSet, combinations);  
}  
  
**int** main() {  
 **int** k = 3;  
 **int** n = 10;  
  
 vector<vector<**int**>> combinations;  
 vector<**int**> set(n);  
 **for** (**int** i = 0; i < n; i++) {  
 set[i] = i + 1;  
 }  
  
 generateCombinations(n, k, set, combinations);  
  
 **for** (**auto** v: combinations) {  
 **for** (**auto** v1: v)  
 cout << v1 << **" "**;  
 cout << **"\n"**;  
 }  
  
}

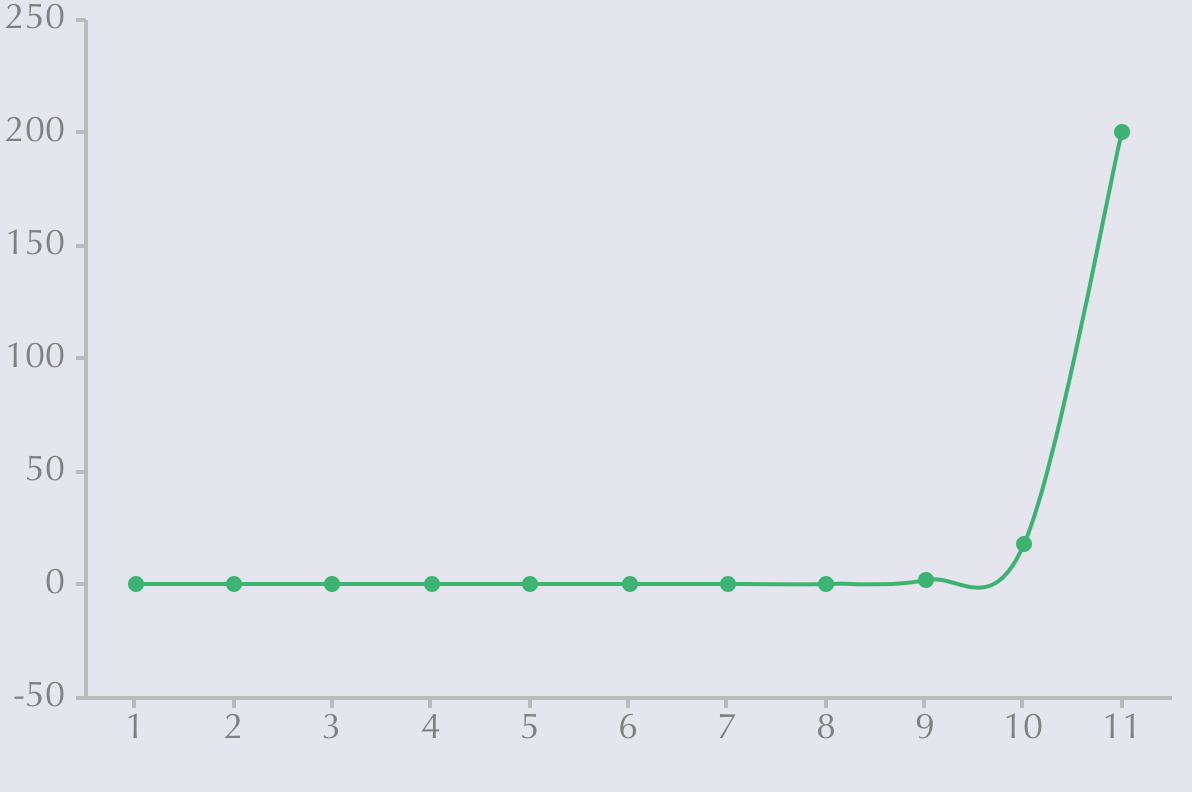
7. Построить графики зависимости количества всех сочетаний из n по k от k при n = (5, 7, 9). 

8. Реализовать алгоритм порождения перестановок.

#include **<iostream>**#include **<utility>**#include **"vector"  
  
using namespace** std;  
  
**void** \_getPermutations(vector<**int**> setAvailable, vector<**int**> setChosen,  
 vector<vector<**int**>> &permutations) {  
 **if** (!setAvailable.empty()) {  
 **for** (**int** j = 0; j < setAvailable.size(); j++) {  
 vector<**int**> setAvailableCopy = setAvailable;  
 vector<**int**> setChosenCopy = setChosen;  
  
 setChosenCopy.push\_back(setAvailableCopy[j]);  
 setAvailableCopy.erase(setAvailableCopy.begin() + j);  
  
 \_getPermutations(setAvailableCopy, setChosenCopy,  
 permutations);  
 }  
 } **else** permutations.push\_back(setChosen);  
}  
  
**void** getPermutations(vector<**int**> initialSet,  
 vector<vector<**int**>> &permutations) {  
 vector<**int**> setChosen;  
 \_getPermutations(initialSet, setChosen, permutations);  
}  
  
**int** main() {  
 **int** setSize = 5;  
  
 vector<**int**> initialSet(setSize);  
 **for** (**int** i = 0; i < setSize; i++) {  
 initialSet[i] = i + 1;  
 }  
  
 vector<vector<**int**>> permutations;  
 getPermutations(initialSet, permutations);  
  
 **for** (**auto** i : permutations) {  
 **for**(**auto** j: i)  
 cout << j << **" "**;  
 cout << **'\n'**;  
 }  
}

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. #include **<iostream>**#include **<utility>**#include **"vector"**#define **TIME\_TEST**(testCode, time) { \  
 clock\_t start\_time = clock(); \  
 testCode \  
clock\_t end\_time = clock();\  
clock\_t sort\_time = end\_time - start\_time; \  
time = (**double**) sort\_time / **CLOCKS\_PER\_SEC**;\  
}  
  
  
**using namespace** std;  
  
**void** \_getPermutations(vector<**int**> setAvailable, vector<**int**> setChosen,  
 vector<vector<**int**>> &permutations) {  
 **if** (!setAvailable.empty()) {  
 **for** (**int** j = 0; j < setAvailable.size(); j++) {  
 vector<**int**> setAvailableCopy = setAvailable;  
 vector<**int**> setChosenCopy = setChosen;  
  
 setChosenCopy.push\_back(setAvailableCopy[j]);  
 setAvailableCopy.erase(setAvailableCopy.begin() + j);  
  
 \_getPermutations(setAvailableCopy, setChosenCopy,  
 permutations);  
 }  
 } **else** permutations.push\_back(setChosen);  
}  
  
**void** getPermutations(vector<**int**> initialSet,  
 vector<vector<**int**>> &permutations) {  
 vector<**int**> setChosen;  
 \_getPermutations(initialSet, setChosen, permutations);  
}  
  
**int** main() {  
 **int** experimentSize = 11;  
  
 vector<**double**> time(experimentSize);  
 **for** (**int** setSize = 1; setSize <= experimentSize; setSize++) {  
 **TIME\_TEST**({  
 vector<**int**> initialSet(setSize);  
 **for** (**int** i = 0; i < setSize; i++) {  
 initialSet[i] = i + 1;  
 }  
  
 vector<vector<**int**>> permutations;  
 getPermutations(initialSet, permutations);  
 }, time[setSize]);  
 }  
  
 **for** (**int** i = 0; i < experimentSize; i++) {  
 cout << i << **" "** << time[i] << **"\n"**;  
 }  
}



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

Полученные показатели были аппроксимированы функцией n!/0.000829

**Тогда на моей ЭВМ:**

* за час будет получено 10 перестановок
* за день будет получено 11 перестановок
* за месяц так же 11 перестановок
* и 12 перестановок за год

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.

**Если ЭВМ в 10 раз мощнее**:

* за час будет получено 11 перестановок
* за день будет получено 12 перестановок
* за месяц так же 12 перестановок
* и 13 перестановок за год

**Если ЭВМ в 10 раз мощнее**:

* за час будет получено 11 перестановок
* за день будет получено 13 перестановок
* за месяц так же 14 перестановок
* и 15 перестановок за год

13. Реализовать алгоритм порождения размещений.

#include **<iostream>**#include **<utility>**#include **"vector"  
  
using namespace** std;  
  
**void** \_getPlacements(**int** k, vector<**int**> setAvailable, vector<**int**> setChosen,  
 vector<vector<**int**>> &permutations) {  
 **if** (setChosen.size() < k) {  
 **for** (**int** j = 0; j < setAvailable.size(); j++) {  
 vector<**int**> setAvailableCopy = setAvailable;  
 vector<**int**> setChosenCopy = setChosen;  
  
 setChosenCopy.push\_back(setAvailableCopy[j]);  
 setAvailableCopy.erase(setAvailableCopy.begin() + j);  
  
 \_getPlacements(k, setAvailableCopy, setChosenCopy,  
 permutations);  
 }  
 } **else** permutations.push\_back(setChosen);  
}  
  
**void** getPlacements(**int** k, vector<**int**> initialSet,  
 vector<vector<**int**>> &permutations) {  
 vector<**int**> setChosen;  
 \_getPlacements(k, initialSet, setChosen, permutations);  
}  
  
**int** main() {  
 **int** setSize = 5;  
 **int** k = 3;  
  
 vector<**int**> initialSet(setSize);  
 **for** (**int** i = 0; i < setSize; i++) {  
 initialSet[i] = i + 1;  
 }  
  
 vector<vector<**int**>> permutations;  
 getPlacements(k, initialSet, permutations);  
  
 **for** (**auto** i : permutations) {  
 **for**(**auto** j: i)  
 cout << j << **" "**;  
 cout << **'\n'**;  
 }  
}

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