Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №4

За 3 семестр

По дисциплине «Дискретная математика»

Тема: «Комбинаторика. Группы.»

Выполнил: студент 2 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Глущенко Т.А.

Брест 2020

**Лабораторная работа №4**

**Комбинаторика. Группы.**

**Комбинаторика.**

***Вариант 1.***

1. Сколько четырехзначных чисел, не превосходящих *6 000*, можно составить, используя только нечетные цифры?
2. Сколькими способами можно выбрать комитет, включающий *6* мужчин и *8* женщин, из группы, состоящей из *12* мужчин и *20* женщин?
3. Пароль на компьютере состоит из шести символов. Первые два из них строчные буквы латинского алфавита (всего *26* букв), а оставшиеся четыре могут быть как цифрами, так и строчными буквами. Сколько можно придумать различных паролей?
4. Решить линейное однородное рекуррентное соотношение *2-го* порядка:  
   
5. Разложить .

***Задание.***

1. *Номер вашего варианта равен остатку от деления вашего порядкового номера по списку на 12. Если ваш порядковый номер кратен 12, выполняете 12 вариант.*
2. *По пункту 4: решить рекуррентное соотношение и программно реализовать вычисление 5 члена числовой последовательности через рекурсию и полученное решение.*
3. *При разложении степени*  *коэффициенты при произведениях* *,* *, рассчитываются по формуле:* *(число перестановок с повторениями) и носят название полиномиальных или мультиномиальных коэффициентов.  
   Например, вычислим коэффициент при произведении*  *в разложении**.  
   Он равен:* *.*
4. *Написать программу генерации перестановок (рекурсивный вариант). Четные номера – реализуют лексикографический порядок генерации. Нечетные номера – реализуют антилексикографический порядок генерации.*
5. *Написать программу генерации k- элементных подмножеств n-элементного множества в лексикографическом порядке. Вход: k – мощность подмножества, n - мощность множества.*

Решение рекуррентного соотношения:

*#include <iostream>*

*#include <vector>*

*void func(std::vector<int> v, int n)*

*{*

*v.push\_back(6 \* v[n - 1] - 9 \* v[n - 2]);*

*n++;*

*if (n == 5) {*

*std::cout << v[n - 1];*

*return;*

*}*

*func(v, n);*

*}*

*int main()*

*{*

*std::vector<int> v = { 3, 21 };*

*int n = 2;*

*func(v, n);*

*}*



Генерация перестановок:

Антилексикографический порядок

#include <iostream>

bool NextSet(int\* a, int n)

{

int j = 1;

while (j != n + 1 && a[j] <= a[j - 1]) {

j++;

}

if (j == n + 1) { //перестановки закончились

return false;

}

int k = 0;

while (a[j] <= a[k]) {

k++;

}

std::swap(a[j], a[k]);

int l = 0, r = j - 1;

while (l < r) { //сортировка оставшейся части последовательности

std::swap(a[l++], a[r--]);

}

return true;

}

void Print(int\* a, int k) //вывод

{

for (int i = 0; i < k; i++) {

std::cout << a[i] << " ";

}

std::cout << std::endl;

}

int main()

{

int n, k;

std::cout << "n = ";

std::cin >> n;

std::cout << "k = ";

std::cin >> k;

int\* a = new int[n];

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

a[i] = i + 1;

}

std::cout << "A bunch of ";

Print(a, n);

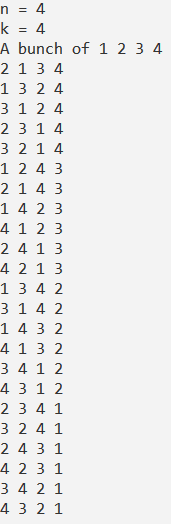
while (NextSet(a, n)) {

Print(a, k);

}

return 0;

}



Генерация в лексикографическом порядке:

#include <iostream>

bool NextSet(int\* a, int n)

{

int j = n - 2;

while (j != -1 && a[j] >= a[j + 1]) {

j--;

}

if (j == -1) { //перестановки закончились

return false;

}

int k = n - 1;

while (a[j] >= a[k]) {

k--;

}

std::swap(a[j], a[k]);

int l = j + 1, r = n - 1;

while (l < r) { //сортировка оставшейся части последовательности

std::swap(a[l++], a[r--]);

}

return true;

}

void Print(int\* a, int k) //вывод

{

for (int i = 0; i < k; i++) {

std::cout << a[i] << " ";

}

std::cout << std::endl;

}

int main()

{

int n, k;

std::cout << "n = ";

std::cin >> n;

std::cout << "k = ";

std::cin >> k;

int\* a = new int[n];

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

a[i] = i + 1;

}

std::cout << "A bunch of ";

Print(a, n);

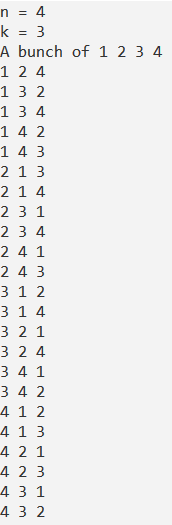
while (NextSet(a, n)) {

Print(a, k);

}

return 0;

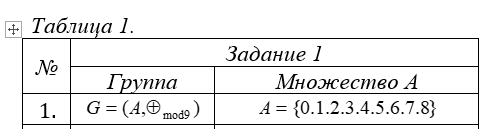
}



**Группы**

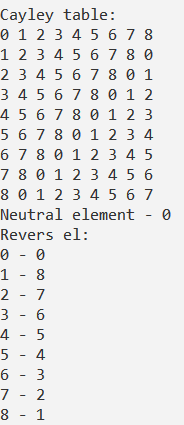
**Задание 1**

**Вариант 1**

****

1. Построить таблицу *Кэли*. Определить *нейтральный* элемент. Для каждого элемента группы указать *обратный* элемент.
2. Определить является ли группа *циклической*, указать *порядок* каждого элемента в группе.
3. Определить является ли группа *абелевой*.
4. Указать все *подгруппы* заданной группы и их порядок. Определить *порождающие элементы* для подгрупп. Для одной из подгрупп построить таблицу *Кэли*.
5. Разложить группу на *смежные* классы по одной из подгрупп. Построить *фактор множество* для группы по данной подгруппе.
6. Построить таблицу *Кэли* для *фактор-группы* по одной из подгрупп.

Пункты *2, 3* выполнять вручную, остальные пункты­ – программная реализация.



Группа является циклической: G = <1> = <2> = <4> = <5> = <7> = <8>

Порядок элементов:

0 – 1

1 – 9

2 – 9

3 – 3

4 – 9

5 – 9

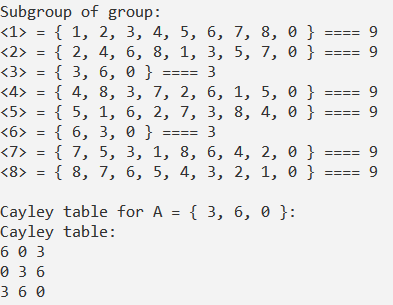
6 – 3

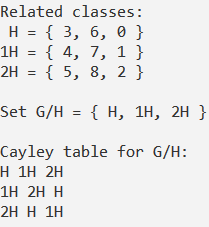
7 – 9

8 – 9

9 – 1

Группа является абелевой





Код программы:

#include <iostream>

#include <vector>

#include <string>

#define op 9 //Число mod

void invers(int\*\* table, int size, int e) //Определяет обратные элементы

{

std::cout << "Revers el: " << std::endl;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

if (table[i][j] == e) {

std::cout << i << " - " << j << std::endl;

break;

}

}

}

}

void Show(int\*\* table, int size) //Выводит таблицу Кэли

{

std::cout << "\nCayley table:" << std::endl;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

std::cout << table[i][j] << " ";

}

std::cout << std::endl;

}

}

int\*\* Cayley(const int\* A, const int size) //Строит таблицу Кэли

{

int e;

bool f = false;

int\*\* table = new int\* [size];

for (int i = 0; i < size; i++) {

table[i] = new int[size];

}

for (int i = 0; i < size; i++) {

if (!f) {

e = i;

}

for (int j = 0; j < size; j++) {

table[i][j] = A[i] + A[j];

table[i][j] == table[i][j] % op;

if (e + table[i][j] == table[i][j] + e && table[i][j] + e == table[i][j]) {

f = true;

}

else {

f = false;

}

}

}

Show(table, size);

std::cout << "Neutral element - " << e << std::endl;

invers(table, size, e);

return table;

}

void subgroup(int size, int\*\* table) //Определяет подгруппы и строит таблицу Кэли для подгруппы

{

std::cout << "\nSubgroup of group: " << std::endl;

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

std::cout << "<" << table[i][j] << "> = { ";

while ((table[i][j] \* n) % op != 0) {

std::cout << (table[i][j] \* n) % op << ", ";

n++;

num++;

}

std::cout << "0 } ==== " << num << std::endl;

}

int A[] = { 3, 6, 0 };

int size\_A = sizeof(A) / sizeof(int);

std::cout << "\nCayley table for A = { 3, 6, 0 }:";

int\*\* table1 = Cayley(A, size\_A);

}

void factor() //Определяет смежные классы, фактор группу

{

int A[] = { 3, 6, 0 }; //Выбранное подмножество

std::vector<std::string> G\_H = { "H" }; //Фактор множество

bool flag = true;

int n = 1; //считает количество смежных классов

std::cout << "\nRelated classes:\n";

std::cout << " H = { 3, 6, 0 }" << std::endl;

while (flag) {

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

if ((A[i] + n) % op == 0) {

flag = false;

}

}

if (flag) {

std::cout << n << "H = { ";

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

std::cout << A[i] + n << ", ";

}

std::cout << "\b\b }" << std::endl;

G\_H.push\_back(std::to\_string(n) + "H");

n++;

}

}

std::vector<int> G; //Файктор множество в переводе на числа

std::cout << "\nSet G/H = { ";

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

std::cout << G\_H[i] << ", ";

G.push\_back(i);

}

std::cout << "\b\b }\n";

std::cout << "\nCayley table for G/H:\n";

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

for (int j = 0; j < n; j++) {

if ((G[i] + G[j]) % n == 0) {

std::cout << "H ";

}

else if (G[i] + G[j] > n) {

std::cout << G\_H[G[i] + G[j] - n] << " ";

}

else {

std::cout << G\_H[G[i] + G[j]] << " ";

}

}

std::cout << std::endl;

}

}

int main()

{

const int A[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8 };

const int size\_A = sizeof(A) / sizeof(int);

int\*\* table1 = Cayley(A, size\_A);

subgroup(size\_A, table1);

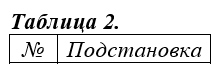
factor();

return 0;

}

**Задание 2**

**Вариант 13**





1. Представить заданную подстановку произведением независимых циклов. Варианты заданий указаны в *таблице 2*. Подстановка из *6* элементов задана в *таблице 2* второй строкой подстановки. Например: подстановка записана в таблице как .
2. Определить порядок *p* заданной подстановки.
3. Составить таблицу *Кэли* для циклической группы, порожденной вашей постановкой. Группа будет иметь вид: . Для вашей подстановки найти обратную подстановку в полученной группе.
4. Определить четность подстановки:  
   – через число транспозиций в разложении подстановки в произведение транспозиций;  
   – через число инверсий в подстановке.
5. Определить *тип* подстановки.
6. Определить является ли ваша группа *абелевой*.
7. Варианты заданий указаны ниже, в *таблице 2*.