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

**1.** Реализовать алгоритм порождения подмножеств.

Программная реализация:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int n;

int s;

void output\_arr (int d[])

{

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

printf ("%d ", d[j]);

printf ("\n");

if (n == 0)

printf("Пустое множество");

puts("");

}

void binary\_vectors (int d[], int i)

{

for (int x = 0; x < 2; x++){

d[i] = x;

if (i == (n - 1)){

++s;

output\_arr (d);

}else binary\_vectors (d, i+1);

}

}

int main()

{

setlocale (LC\_ALL, "rus");

printf ("Введте размер множества: ");

scanf ("%d", &n);

int i=0;

int d[n];

s = 0;

binary\_vectors(d, i);

printf ("Количество подмножеств: %d", s);

}

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

Рис.1 - График зависимости количества всех подмножеств от мощности множества

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

Рис.2 - График зависимости времени выполнения алгоритма п.1 на 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.

Максимальная мощность множества, для которого можно получить все подмножества не более чем за час на ЭВМ при , :

Максимальная мощность множества, для которого можно получить все подмножества не более чем за сутки на ЭВМ при , :

Максимальная мощность множества, для которого можно получить все подмножества не более чем за месяц на ЭВМ при , :

Максимальная мощность множества, для которого можно получить все подмножества не более чем за год на ЭВМ при , :

**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:

В 100 раз быстрее:

Максимальная мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ в 10 раз быстрее при, :

Час:

Сутки:

Месяц:

Год:

Максимальная мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ в 100 раз быстрее при, :

Час:

Сутки:

Месяц:

Год:

**6.** Реализовать алгоритм порождения сочетаний.

Программная реализация:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int n;

int k;

int s;

void output\_set (int c[])

{

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

printf ("%d ", c[j]);

printf ("\n");

if (n == 0)

printf("Пустое множество");

puts("");

}

void combination (int c[], int i, int b)

{

for (int x = b; x <=(n-k+i+1); x++){

c[i] = x;

if (i == (k-1)){

++s;

output\_set (c);

}else combination (c, i+1, x+1);

}

}

int main()

{

setlocale (LC\_ALL, "rus");

printf ("Введте размер множества: ");

scanf ("%d", &n);

printf ("Введте количество элеменов в сочетании: ");

scanf ("%d", &k);

int i=0;

int b=1;

int c[k];

s = 0;

combination(c, i, b);

printf ("Количество сочетаний: %d", s);

}

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

Рис. 3.1 - График зависимости количества всех сочетаний из n по k от k при

n = 5

Рис. 3.2 - График зависимости количества всех сочетаний из n по k от k при

n = 7

Рис. 3.3 - График зависимости количества всех сочетаний из n по k от k при

n = 9

**8.** Реализовать алгоритм порождения перестановок.

Программная реализация:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int n;

int s;

void set\_init(int m[])

{

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

m[i] = i + 1;

}

void output\_set (int p[])

{

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

printf("%d ", p[i]);

if (n == 0)

printf("Пустое множество");

puts("");

}

int resid(int m[], int n, int b[], int x)

{

int len\_b = 0;

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

if (m[i] != x){

b[len\_b] = m[i];

++len\_b;

}

return len\_b;

}

void permutation(int p[], int i, int m[], int t)

{

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

p[i] = m[j];

if (i == n-1){

++s;

output\_set(p);

}else{

int b[t-1], len\_b;

len\_b = resid(m, t, b, p[i]);

permutation(p, i+1, b, len\_b);

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

printf("Введите мощность множества: ");

scanf("%d", &n);

int m[n];

set\_init(m);

int p[n];

int i = 0;

int t = n;

s = 0;

permutation(p, i, m, t);

printf("Количество перестановок: %d\n", s);

}

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

Рис. 4 - График зависимости количества всех перестановок от мощности множества

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

Рис. 5 - График зависимости времени выполнения алгоритма п.8 на 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.

Максимальная мощность множества, для которого можно получить все перестановки не более чем за час при , :

Максимальная мощность множества, для которого можно получить все перестановки не более чем за сутки при, :

Максимальная мощность множества, для которого можно получить все перестановки не более чем за месяц при, :

Максимальная мощность множества, для которого можно получить все перестановки не более чем за год при, :

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

В 10 раз быстрее:

В 100 раз быстрее:

Максимальная мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на ЭВМ, в 10 раз быстрее при, :

Час:

Сутки:

Месяц:

Год:

Максимальная мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на ЭВМ, в 10 раз быстрее при, :

Час:

Сутки:

Месяц:

Год:

13. Реализовать алгоритм порождения размещений.

Программная реализация:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int n;

int k;

int s;

void set\_init(int m[])

{

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

m[i] = i + 1;

}

void output\_set (int a[])

{

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

printf("%d ", a[i]);

if (n == 0)

printf("Пустое множество");

puts("");

}

int resid(int m[], int n, int b[], int x)

{

int len\_b = 0;

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

if (m[i] != x){

b[len\_b] = m[i];

++len\_b;

}

return len\_b;

}

void arrangement(int a[], int i, int m[], int t)

{

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

a[i] = m[j];

if (i == k-1){

++s;

output\_set(a);

}else{

int b[t-1], len\_b;

len\_b = resid(m, t, b, a[i]);

arrangement(a, i+1, b, len\_b);

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

printf("Введите мощность множества: ");

scanf("%d", &n);

printf ("Введите колличество мест для размещения: ");

scanf("%d", &k);

int m[n];

set\_init(m);

int a[k];

int i = 0;

int t = n;

s = 0;

arrangement(a, i, m, t);

printf("Количество размещений: %d\n", s);

}

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

Рис. 6.1 - График зависимости количества всех размещений из n по k от k при n = 5

Рис. 6.2 - График зависимости количества всех размещений из n по k от k при n = 7

Рис. 6.3 - График зависимости количества всех размещений из n по k от k при n = 7