**Лабораторная работа № 2**

студента группы ИТ-211

Хлыбова Кирилла Сергеевича

Выполнение:\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_

**СОЧЕТАНИЯ С ПОВТОРЕНИЯМИ И БЕЗ ПОВТОРЕНИЙ**

Цель работы: приобрести навыки по работе с различными видами сочетаний.

**Содержание работы**

1) Разработать алгоритм нахождения сочетаний в заданном множестве.

2) Построить блок-схему алгоритма.

3) Написать программу для нахождения сочетаний.

4) Составить отчет о проделанной работе.

**Ход работы**

Разработал алгоритм нахождения сочетаний согласно заданию.

Спецификации функций:

**1) def factorial(n)**

Назначение – нахождение факториала числа.

Привожу блок схему алгоритма (Рис. 1):



Рис. 1 Блок-схема ф-ии factorial(n)

Код функции:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **def** **factorial**(n):  f = **1**  **while** n > **1**:  f = f \* n  n -= **1**  **return** f |

**2) def countWithoutDouble(n, m)**

Назначение – нахождение количества сочетаний без повторений.

Код функции:

|  |  |
| --- | --- |
| 1  2 | **def** **countWithoutDouble**(n, m):  **return** factorial(n) / (factorial(m) \* factorial(n - m)) |

**3) def countWithDouble(n, k)**

Назначение – нахождение количества сочетаний с повторениями.

Код функции:

|  |  |
| --- | --- |
| 1  2 | **def** **countWithDouble**(n, k):  **return** factorial(n + k - **1**) / (factorial(k) \* factorial(n - (k - **1**))) |

**4) def printWithout(m, n)**

Назначение – вывод сочетаний без повторений.

Привожу блок-схему алгоритма (Рис. 2):



Рис. 2 Блок-схема ф-ии printWithout(m, n)

Код-функции:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **def** **printWithout**(m, n):  arr = []  **for** i **in** range(m):  arr.append(i)  arr.append(n)  arr.append(**0**)  **while** **True**:  print(arr[**0**:m])  **for** j **in** range(len(arr)-**1**):  **if** arr[j] + **1** == arr[j+**1**]:  arr[j] = j  **else**:  **break**  **if** j < m:  arr[j] = arr[j] + **1**  **else**:  **break**  **return** arr |

**5)** **def printWith(m, k)**

Назначение – вывод сочетаний с повторениями.

Привожу блок-схему алгоритма (Рис. 3):



Рис. 3 Блок-схема ф-ии def printWith(m, k)

Код функции:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **def** **printWith**(m, k):  some = [**0**]\*k  **while** some **is** **not** **None**:  **yield** some  some = (generSome(some, m))  **return** **None** |

**6)** **def generSome(some, m)**

Назначение – генерация новых сочетаний.

Привожу блок-схему алгоритма (Рис. 4):



Рис. 4 Блок-схема ф-ии def generSome(some, m)

Код функции:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | **def** **generSome**(some, m):  k = len(some)  **for** i **in** range(k – **1**, -**1**, -**1**):  **if** some[i] < m – **1**:  **break**  **else**:  **return** **None**  some[i] = some[i] + **1**  **for** j **in** range(i+**1**, k):  some[j] = some[i]  **return** some |

Остальной код программы (без функций):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | print("Введите 0, если хотите вывести сочетания без повторений")  print("Введите 1, если хотите вывести сочетания с повторениями")  check = int(input())  **if** check == **0**:  print("из скольки элементов?")  n = int(input())  print("по сколько?")  m = int(input())  print("Количество без повторений")  printWithout(m, n)  print(countWithoutDouble(n, m))  **else**:  print("из скольки элементов?")  m = int(input())  print("по сколько?")  k = int(input())  print("Количество с повторениями")  **for** s **in** printWith(m, k):  print(s)  print(countWithDouble(m, k)) |

**Вывод**: в ходе выполнения лабораторной работы получил дополнительные знания по работе с сочетаниями и реализации алгоритма комбинаторики в виде программы.