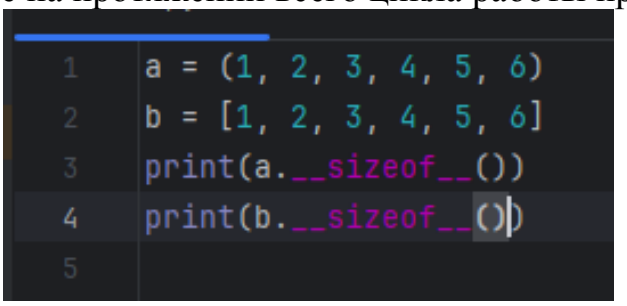
ЛАБОРАТОРНАЯ РАБОТА 6.
КОРТЕЖИ, СЛОВАРИ И МНОЖЕСТВА. РАБОТА С ФУНКЦИЯМИ

6.1. СПРАВОЧНЫЙ МАТЕРИАЛ

Кортеж — это неизменяемый список. Кортеж не может быть изменён никаким способом после его создания. Кортеж определяется так же, как список, за исключением того, что набор элементов заключается в **круглые скобки**, а не в квадратные.

Так же как списки они могут состоять из элементов разных типов, перечисленных через запятую. Кортежи защищены от изменений (в случае преднамеренного или случайного изменения) и имеет меньший размер. Таким образом, каждый кортеж создается только один раз и существует в первоначальном виде на протяжении всего цикла работы приложения.



Результат работы:
72
88.
Работать с кортежами можно, как со списками.

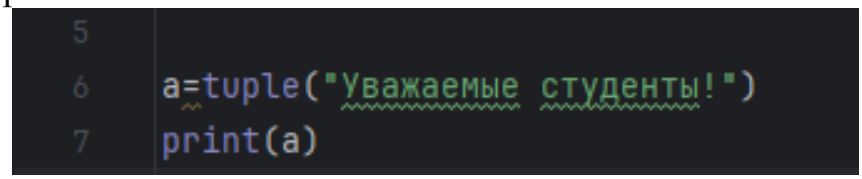
Создаем пустой кортеж:

```
a = tuple() # С помощью встроенной функции tuple()

a = () # С помощью литерала кортеж а

a = ('s', ) # запятая показывает, что это кортеж, а не строка
```

Например:



После выполнения команды создается кортеж:
('У', 'в', 'а', 'ж', 'а', 'е', 'м', 'ы', 'е', ' ', 'с', 'т', 'у', 'д', 'е', 'н', 'т', 'ы', '!')

Операции с кортежем такие же, как и методы работы со списками, что касается неизменяемых методов.

При необходимости, мы можем разложить кортеж на отдельные переменные для дальнейшей работы с ними:

```
1.  data = ["Иван", 37, "Яндекс"]
2.  name, age, company = tuple(data)
3.  print(name) # Иван
4.  print(age) #37
5.  print(company) # Яндекс
```

Таким образом, благодаря поддержке кортежей, Python обеспечивает разработчика гарантией защиты данных от непреднамеренных изменений в приложении, что повышает его отказоустойчивость. Это делает Python универсальным и надежным средством для работы с производительными приложениями, а также с Big Data обработкой.

Для оператора + *типы list и tuple не являются совместимыми*. Для соединения объектов двух разных типов требуется явное преобразование.

Словари

Словарь это ассоциативный массив или хеш, то есть коллекция содержащая пары "ключ: значение". Класс словаря **dict**. Словарь имеет свойства:

✓ перечислимость (enumerable)

✓ изменяемость (mutable)

Словарь это не последовательность. Словарь подобен списку, но доступ к его элементам осуществляется не по индексу, а по ключу. Ключ должен иметь тип, допускающий вычисление хешфункции, то есть обладать свойством хешируемости (hasheable).

Хеш-функция, это функция, параметром которой является объект, а результатом целое число, служащее индексом ячейки памяти во внутреннем массиве, где будет размещено связанное с объектом значение, для которого объект является ключом: это упрощенная модель, реальные ассоциативные массивы реализованы на базе сложных высокооптимизированных алгоритмов.

Только неизменяемый объект хешируем: если объект-ключ изменится, изменится и его индекс во внутреннем массиве, и он не сможет быть найден в словаре.

Композитный объект хешируем только в том случае, если все его элементы хешируемы.

Создание объекта

Литерал типов *dict* и *set* это последовательность элементов, разделенных запятыми и заключенных в фигурные скобки.

```
d = { 'name': 'abcd', 'f1': 14, 5: 27.12, 'pair': [ 1, 2 ] }
d = { 'name': 'abcd', 'f1': 14, 5: 27.12, [ 1, 2 ]: 'pair' } # Ошибка
```

Доступ к элементам осуществляется так же, как для списков, но операндом является ключ, а не индекс.

Методы словарей

- Преобразование словаря в итерируемый объект. При итерации по коллекции-результату порядок элементов не определен.

```
d.keys()   # Возвращает коллекцию ключей
d.values() # Возвращает коллекцию значений
d.items()  # Возвращает коллекцию пар (ключ, значение)
```

- Добавление элементов в словарь

```
d.update(other_dict) # Добавить элементы из другого словаря
d.update(color='Green', count=27) # или из переданных параметров
```

Множества

Множества в Python – это структура данных, которые содержат неупорядоченные элементы. Элементы также не является индексированным. Как и список, множество позволяет внесение и удаление элементов. Однако, есть ряд особенных характеристик, которые определяют и отделяют множество от других структур данных:

- ✓ Множество не содержит дубликаты элементов;
- ✓ Элементы множества являются неизменными (их нельзя менять), однако само по себе множество является изменяемым, и его можно менять;
- ✓ Так как элементы не индексируются, множества не поддерживают никаких операций **среза** и **индексирования.**

Множество - это частный случай словаря, где ключ сам по себе является значением.

Функции в Python

Функция – особым образом, сгруппированный набор команд, которые выполняются последовательно, но воспринимаются как единое целое. При этом функция может возвращать (или не возвращать) свой результат.

Создание функции

Для того чтобы использовать какую-нибудь собственную функцию, вначале необходимо ее объявить (создать).

Блок функции начинается с ключевого слова **def**, после которого следуют название функции и круглые скобки ( ). Любые аргументы, которые принимает функция, должны находиться внутри этих скобок. После скобок идет двоеточие и с новой строки с отступом начинается тело функции.

def <имя функции>(аргументы):
<тело функции>

После создания функции, ее можно исполнять, вызывая из другой функции или напрямую из оболочки Python. Для вызова функции следует ввести ее имя и добавить в скобках аргументы.

<имя функции>(аргументы)

Обращение к ранее объявленной функции с целью выполнения ее команд называется ***вызовом***.

Имена функций должны состоять из маленьких букв, а слова разделяться символами подчеркивания. Аргументы (параметры) могут изменять поведение функции.

Аргументы функции в Python

Вызывая функцию, мы можем передавать ей следующие типы аргументов:

1. Обязательные аргументы (Required arguments)
2. Аргументы-ключевые слова (Keyword argument)
3. Аргументы по умолчанию (Default argument)
4. Аргументы произвольной длины (Variable-length argumens)

Если при создании функции мы указали количество передаваемых ей аргументов и их порядок, то и вызывать ее мы должны с тем же количеством аргументов, заданных в нужном порядке.

Оператор return

Для возвращения результата из функции используется оператор **return**. Функции могут возвращать кортежи, в таком случае после оператора **return** надо писать возвращаемые значения через запятую. В общем случае функция может не содержать оператор **return**, значит функция просто выполняет действия и ничего не возвращает. Иногда такие функции называют ***процедурами***.

Рассмотрим пример описания функции и ее последующего вызова.

```
""" arg2 и arg3 – необязательные аргументы, принимают значение,
объявленное по умолчанию, если не задать им другое значение при вызове
функции.
"""
def myfunction(arg1, arg2 = 100, arg3 = 'test.txt'):
  return arg3, arg2, arg1
""" Функция вызывается со значением первого аргумента – "Argument 1",
второго – по умолчанию, и третьего – "Named argument".
"""
ret1, ret2, ret3 = myfunction('Первый аргумент', arg3 = 'Третий
аргумент')
""" ret1, ret2 и ret3 принимают значения "Третий аргумент", 100,
"Первый аргумент" соответственно
"""
print (ret1, ret2, ret3)
#Результат:   Третий аргумент 100 Первый аргумент
```

lambda-функция

В Python существует возможность создавать анонимные функции и встраивать их сразу в выражения. Для этого используются lambda-функции. Синтаксис определения lambda-функции следующий:

lambda *аргументы* : *тело_функции* (*передаваемые_аргументы*)

Приведем пример:

```
# Следующая запись эквивалентна def f(x): return x + 1

functionvar = lambda x: x + 1
print (functionvar(5))
#6

import math
pi_const = round(math.pi, 2) #округляем pi для последующих вычислений
#lambda-функция для вычисления площади круга
s_krug = lambda radius: pi_const*(radius**2)
print (s_krug(5))
print (s_krug(12))
#78.5
#452.16
```

Переменное количество параметров в функции

Когда число параметров заранее неизвестно, тогда в список параметров функции в качестве одного из параметров добавляется *args. Звёздочка (*) обозначает, что данный параметр ожидает кортеж.

```
def calculate_sum(a, *args):
  sum = a
  for i in args:
    sum += i
  return sum

print(calculate_sum(10)) # 10
print(calculate_sum(10, 11, 12)) # 33
print(calculate_sum(1, 2, 94, 6, 2, 8, 9, 20, 43, 2)) # 187
```

Так же можно передавать в качестве параметра словарь, для этого указывается 2 звездочки **, например, **kwargs.

```
def print_names(f1, l1, **kwargs):
  print(f1, l1, end=' ')
  for key in kwargs:
    print(key, kwargs[key], end=' ')

print_names("Андрей", "Алексей")
print_names("Василий", "Петр", alex="Алексей", leon="Леон")
# Андрей Алексей Василий Петр alex Алексей leon Леон


def named_infinity(**kwargs):
    print(kwargs)

named_infinity(first='nothing', second='else', third='matters')
```

```
# {'first': 'nothing', 'second': 'else', 'third': 'matters'}
```

В Python допускаются вложенные и рекурсивные функции. Функции, которые объявляются и вызываются внутри других функций, называются *вложенными*. А функции, которые вызывают сами себя – *рекурсивными*.

Иногда в функции необходимо вернуть множество значений, но количество возвращаемых значений заранее неизвестно. В этом случае можно возвратить два параметра – количество значений и сам список значений, например:

```
def odd_numbers(x):
  l=[]
  for i in range(x+1):
    if (i%2)!=0:
      l.append(i)
  return len(l), l

no_of_odds, odd_list = odd_numbers(100)
```

Аннотация типов в функциях

Python — язык с динамической типизацией. По этой причине вполне возможны ситуации, когда вопреки ожиданиям разработчика в функцию подаются, например, не целые числа, а, допустим, строки, что вызовет ошибку. Чтобы отслеживать подобные случаи и сильнее контролировать процесс выполнения программы, используется аннотация типов.

С помощью аннотации типов указывается, что параметры в функции имеют строго определенный тип, например.

```
def mult (a: int, b: int) -> int:
    return a * b
```

В этой функции мы показали, что аргументы и результат должны быть целыми. Если передать float, то функция выполнится как обычно, однако IDE предупредит нас, что было получено неожиданное значение.

Пустая функция

Иногда разработчики оставляют реализацию на потом, и чтобы объявленная функция не генерировала ошибки из-за отсутствия тела, в качестве заглушки используется ключевое слово **pass**:

```
def empty():
    pass
```

Изменяемые и неизменяемые параметры функций

Передаваемые в функцию параметры могут изменяться внутри функции, однако при возврате из функции не все измененные параметры сохранят свое значение. Это зависит от того, какие виды объектов передаются в функцию. Различают неизменяемые объекты и изменяемые.

Если объект неизменяемый, то он передаётся в функцию по значению. Неизменяемые объекты это:

- Числовые типы (int, float, complex).
- Строки (str).
- Кортежи (tuple).

Чтобы вы ни делали внутри функции с этими параметрами – вы работаете лишь с локальной его копией и при возврате из функции значение такого параметра останется прежним.

Изменяемые объекты передаются в функцию по ссылке. Изменяемыми они называются потому что их содержимое можно менять внутри функции, при этом ссылка на сам объект остается неизменной.

В Python изменяемые объекты это:

- Списки (list).
- Множества (set).
- Словари (dict).

Рекурсивные функции

Под рекурсивным определением функции или под рекурсией будем понимать такое описание (представление) функции, которое содержит внутри себя ссылку на определяемую функцию. Другими словами, функция из своего тела осуществляет вызов непосредственно самой себя (***прямая рекурсия***) с другими аргументами (параметрами), либо вызывает саму себя из других функций или процедур, находящихся внутри тела рекурсивной функции (***косвенная рекурсия***).

Важнейшим признаком любой рекурсивной функции является ***условие выхода из рекурсии.*** Такое условие выхода содержится, как правило, в самом начале рекурсивных функций и определяет, когда рекурсия должна быть закончена

Пусть **P** – рекурсивная функция, тогда выполнение действий в рекурсивной функции может быть организовано одним из следующих способов:

Рекурсивный подъём	Рекурсивный спуск	Рекурсивный спуск и рекурсивный подъём

`def P():`
`  P()`
`  операторы`
	`def P():`
	`  операторы`
	`  P()`
		`def P():`
		`  операторы`
		`  P()`
		`  операторы`

Приведем примеры рекурсивных функций суммы, разности натуральных чисел, а также функции вычисления факториала числа:

```
def SumXY ( x, y ):
  if (x == 0):# условие выхода из рекурсии
    return(y)
  else:
    return(SumXY(x-1,y)+1) #вызываем рекурсивно сумму с меньшим целым

def SubXY ( x, y ):
  if (y == 0):# условие выхода из рекурсии
    return(x)
  else:
    return(SubXY(x,y-1)-1)

def Factorial ( n ):
  if (n == 1):# условие выхода из рекурсии
    return(1)
  else:
    return(Factorial(n-1)*n)

print(SumXY(5,10))
print(SubXY(24,11))
print(Factorial(5))
```

Результат:

```
15
13
120
```

6.2. ПРИМЕРЫ РЕШЕНИЯ ЗАДАЧ

Кортежи

Пример 1. Явное преобразование:

```
m = [ 'abcdef', 14, 27.12, [ 1, 2 ] ]
c = ( 'abcdef', 14, 27.12, [ 1, 2 ] )
m2 = m + m         # Правильно
m2 = m + c         # Ошибка
m2 = m + list(c)   # Правильно, результат list
c2 = c + tuple(m) # Правильно, результат tuple
```

Пример 2. В кортеж может быть помещен объект изменяемого композитного типа. Такой объект *остается изменяемым*.

```
a = 18
b = 'b_string'
c = [ 1, 2 ]
y = [ 'abcdef', a, b, c ]
print(y) # => [ 'abcdef', 18, 'b_string', [1, 2] ]
a = 33
b = 'b_modified'
c[1] = 433
c = [ 8, 9 ]
print(y) # => [ 'abcdef', 18, 'b_string', [1, 433] ]
```

В результате всех операций измененным в кортеже или списке оказался только объект изменяемого композитного типа.

Словари

Пример 3. Создать словарь.

```
d={"type":"Creta", 5:77777,"Name":"Иван"}
print(d[5])
print(d["Name"])
```

Множества

Пример 4. Создание множества из списка. Это можно сделать, вызвав встроенную функцию Python под названием set():

```
num_set = set([1, 2, 3, 4, 5, 6])
print(num_set)
```

Результат:

```
{1, 2, 3, 4, 5, 6}
```

Функции

Пример 5. Создание процедуры (т.к. нет оператора return) вычисления квадрата числа (обязательные аргументы).

```
def primer(n):
    print(n**2)
primer(7)
```

Пример 6. Создание процедуры вычисления наибольшего числа (обязательные аргументы).

```
def bigger(a,b):
    if a > b:
        print(a)
    else:
        print(b)
bigger(10,12)
```

Пример 7. Использование процедуры с ключевыми словами (аргументы - ключевые слова используются при вызове функции. Благодаря ключевым аргументам, вы можете задавать произвольный (то есть не такой, каким он описан при создании функции) порядок аргументов.

```
def person(name, age):
    print("Возраст", name, age, "лет")
person(age=23, name="Елена")
```

Пример 8. Создание функции (есть оператор **return)**. Выражение **return** прекращает выполнение функции и возвращает указанное после выражения значение. Соответственно, теперь становится возможным, например, присваивать результат выполнения функции какой-либо переменной.

```
def bigger(a,b):
    if a > b:
        return(a)     # Если а больше чем b, то возвращаем а и
                    #прекращаем выполнение функции
    return b         # Незачем использовать else.
num = bigger(40,50)
print(num)
```

Пример 9. Функция поиска минимального элемента в массиве

```
def GetMin(array):
  minval = array[0] #минимальное значение в массиве
  MinIndex=0 # MinIndex - индекс минимального элемента в массиве
  for i in range (1,len(array)):
    if (array[i] < minval):
      minval = array[i]
      MinIndex=i
  return minval, MinIndex;

numbers = [ -3, 322, 16, 27, -55, 43, 2, 34, 0]
minval, minind = GetMin(numbers)# вызов такой функции
print(f'Минимальный элемент в массиве {minval}, его индекс = {minind}')
```

Результат:

```
[evaluate minArray.py]
Минимальный элемент в массиве -55, его индекс = 4
```

6.3. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

ОБЩИЕ ЗАДАНИЯ ДЛЯ ВСЕХ ВАРИАНТОВ

Задание 1. Используя операции индексирования и среза выведите на экран первый и третий элементы кортежа (1, 2, 3, 4, 5), а также срез кортежа из первых трех элементов. Реализуйте вывод двумя способами: используя только положительные и только отрицательные индексы.

Задание 2. Дан кортеж ((1, 2, ('Ok!', 3)), ('tuple', 4), 5). Выведите на экран строку 'Ok!', использовав синтаксис доступа к элементу кортежа по его индексу.

Задание 3. Дан кортеж (['кит', 1, 3], 5). Замените в списке 'кит' на 'кот', удалите единицу, а также измените значение последнего элемента списка, возведя тройку в квадрат. Выведите кортеж на экран. Попробуйте изменить второй элемент кортежа, умножив его на два.

Задание 4. Создайте словарь содержащий данные о человеке. В качестве строковых ключей используйте его имя, возраст и пол. Значения придумайте самостоятельно. Выведите полученный словарь на экран.

Задание 5. Дан словарь d = {'1': 1.29, '2': 0.43}. Используя доступ к элементам словаря по ключу, найдите произведение 1.29*0.43, после чего добавьте результат в словарь, а затем выведите значение нового элемента на экран.

Задание 6. Сколько элементов будет содержать словарь d_1 = {'1': 1, '2': 2} после добавления к нему элементов словаря d_2 = {'2': 'два', '3': 3}? Каковы будут значения элементов итогового словаря? Проверьте свой ответ программно.

Задание 7 (множества). Даны три слова 'аквариум', 'мармелад' и 'рама'. Выведите на экран сперва все виды букв, которые присутствуют во всех словах сразу, а затем все виды букв, которые присутствуют в любом из них.

Задание 8. Напишите две функции S(r) и l(r), принимающие в качестве аргумента радиус окружности и возвращающие площадь круга и длину этой окружности соответственно. Затем напишите функцию krug(), которая спрашивает у пользователя радиус окружности, а затем при помощи функций S(r) и l(r) выводит на экран площадь круга и длину окружности, разделённые пробелом.

ЗАДАЧИ ДЛЯ РАЗЛИЧНЫХ ВАРИАНТОВ

ВНИМАНИЕ! Во всех задачах нужно разработать собственные функции и использовать их. Данные должны вводиться и результаты выводиться на консоль

№ 1. Треугольник задан координатами своих вершин. Составить функцию вычисления его площади.

№ 2. Составить функцию нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел (НОК(A,B)= $\frac{A*B}{НОД(A,B)}$).

№ 3. Составить функцию нахождения наибольшего общего делителя четырех натуральных чисел.

№ 4. Составить функцию нахождения наименьшего общего кратного трех натуральных чисел.

№ 5. Написать функцию нахождения суммы большего и меньшего из 3 чисел.

№ 6. Вычислить площадь правильного шестиугольника со стороной a, используя функцию вычисления площади треугольника.

№ 7. На плоскости заданы своими координатами n точек. Составить функцию, определяющую между какими из пар точек самое большое расстояние. Указание. Координаты точек занести в массив.

№ 8. Проверить, являются ли данные три числа взаимно простыми. Написать функцию с тремя параметрами.

№ 9. Написать функцию вычисления суммы факториалов всех нечетных чисел от 1 до 9.

№ 10. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ (A, B, C, D — натуральные числа). Составить функцию:

- деления дроби на дробь;
- умножения дроби на дробь;
- сложения этих дробей.

Ответ должен быть несократимой дробью.

№ 11. На плоскости заданы своими координатами n точек. Создать матрицу, элементами которой являются расстояние между каждой парой точек.

№ 12. Написать функцию, в которую передается одномерный массив и целое число k. Функция возвращает новый массив, элементы которого циклически сдвинуть на k мест вправо.

№ 13. Написать функцию, которая выдает сформированный массив X(N), N-й член которого определяется формулой X(N) = $\frac{1}{N!}$.

№ 14. Составить функцию вычисления суммы факториалов всех четных чисел от m до n.

№ 15. Заменить отрицательные элементы линейного массива их модулями, не пользуясь стандартной функцией вычисления модуля. Подсчитать количество произведенных замен. Использовать функции.

№ 16. Дан массив A(N). Сформировать массив B(M), элементами которого являются большие из двух рядом стоящих в массиве A чисел. (Например, массив A состоит из элементов 1, 3, 5, -2, 0, 4, 0. Элементами массива B будут 3, 5, 4.). Использовать функции.

№ 17. Дан массив A(N) (N — четное). Сформировать массив B(M), элементами которого являются средние арифметические соседних пар рядом стоящих в массиве A чисел. (Например, массив A состоит из элементов 1, 3, 5, -2, 0, 4, 0, 3. Элементами массива B будут 2; 1,5; 2; 1,5.)

№ 18. Дано простое число. Составить функцию, которая будет находить следующее за ним простое число.

№ 19. Составить функцию для нахождения наименьшего нечетного натурального делителя k ($k \neq 1$) любого заданного натурального числа n.

№ 20. Дано натуральное число N. Составить функцию формирования массива, элементами которого являются цифры числа N.

№ 21. Составить функцию, определяющую, в каком из данных двух чисел больше цифр.

№ 22. Написать функцию, которая заменяет натуральное число на число, которое получается из исходного записью его цифр в обратном порядке (например, дано число 156, нужно получить 651).

№ 23. Даны натуральные числа K и N. Составить функцию формирования массива A, элементами которого являются числа, сумма цифр которых равна K и которые не больше N.

№ 24. Даны три квадратных матрицы A, B, C n-го порядка. Вывести на экран ту из них, норма которой наименьшая. Нормой матрицы считать максимум из абсолютных величин ее элементов. Использовать функции.

№ 25. Два натуральных числа называются «дружественными», если каждое из них равно сумме всех делителей (кроме его самого) другого (например, числа 220 и 284). Найти все пары «дружественных чисел», которые не больше данного числа N.

№ 26. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов» из отрезка [n, 2n], где n — заданное натуральное число больше 2.

№ 27. Натуральное число, в записи которого n цифр, называется числом Амстронга, если сумма его цифр, возведенная в степень n, равна самому числу. Найти все эти числа от 1 до k.

№ 28. Написать функцию, которая находит и выводит на печать все четырехзначные числа вида $\overline{abcd}$, для которых выполняется: a, b, c, d — разные цифры; б) $\overline{ab} - \overline{cd}$ =a+b+c+d.

№ 29. Найти все простые натуральные числа, не превосходящие n, двоичная запись которых представляет собой палиндром, т.е. читается одинаково слева направо и справа налево.

№ 30. Найти все натуральные n-значные числа, цифры в которых образуют строго возрастающую последовательность (например, 1234, 5789).

№ 31. Найти все натуральные числа, не превосходящие заданного n, которые

делятся на каждую из своих цифр.

№ 32. Составить программу для нахождения чисел из интервала [M; N], имеющих наибольшее количество делителей.

№ 33. Дано натуральное число n. Выяснить, можно ли представить n в виде произведения трех последовательных натуральных чисел.

№ 34. На части катушки с автобусными билетами номера шестизначные. Составить программу, определяющую количество счастливых билетов на катушке, если меньший номер билета — N, больший — M (билет является счастливым, если сумма первых трех его цифр равна сумме последних трех).

№ 35. Написать функцию, определяющую сумму n-значных чисел, содержащих только нечетные цифры. Определить также, сколько четных цифр в найденной сумме.

№ 36. Составить функцию разложения данного натурального числа на простые множители. Например, $200 = 2^3 \cdot 5^2$.

№ 37. Дано натуральное число n. Найти все меньшие n числа Мерсена. (Простое число называется числом Мерсена, если оно может быть представлено в виде $2^p - 1$, где p – тоже простое число. Например, 31 = 25 - 1 – число Мерсена.)

№ 38. Даны два одномерных массива A и B. Найти сумму их минимальных и сумму их максимальных элементов, используя две функции: поиск минимального элемента и поиск максимального элемента вектора.

№ 39. Составить функцию, позволяющую определить позицию самого правого вхождения заданного символа в исходный массив строк. Если строка не содержит символа, результатом работы функции является значение «–1».

№ 40. В функцию передается одномерный массив из n элементов. Функция «переворачивает» этот массив и возвращает новый массив, в котором первым элементом будет последний элемент переданного массива, вторым – предпоследний и т.д.

№ 41. Найдите сумму цифр заданного натурального числа. Использовать рекурсию.

№ 42. Подсчитать количество цифр в заданном натуральном числе. Использовать рекурсию.

№ 43. Составить рекурсивную функцию вычисления НОД двух натуральных чисел.

№ 44. Составить рекурсивную функцию нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке. Например, для числа 1234 получаем ответ 4321.

№ 45. Составить рекурсивную функцию перевода данного натурального числа в p-ичную систему счисления ($2 \le p \le 9$).

№ 46. Написать рекурсивные функции вычисления суммы, разности, произведения, целочисленного деления и вычисления остатка от деления двух натуральных чисел.

№ 47. Составить программу вычисления суммы, использовать рекурсию:

1! + 2! + 3! + ... +n! (n<15).

Примечание. Тип результата значения функции long.

№ 48. Составить программу вычисления суммы: 2! + 4! + 6! + ... + n! (n<16, n — четное). Примечание. Использовать рекурсию. Тип результата значения функции long.

№ 49. Написать рекурсивную функцию, выдающую n-ое число Фибоначчи.

№ 50. С помощью рекурсивной функции вычислить сумму элементов одномерного массива, который предварительно заполняется случайным образом с помощью стандартных функций генерации случайных чисел.

РАСПРЕДЕЛЕНИЕ ЗАДАЧ ПО ВАРИАНТАМ

№ задачи	Вариант 1	Вариант 2	Вариант 3	Вариант 4	Вариант 5	Вариант 6	Вариант 7	Вариант 8	Вариант 9	Вариант 10	Вариант 11	Вариант 12	Вариант 13	Вариант 14	Вариант 15	Вариант 16	Вариант 17	Вариант 18	Вариант 19	Вариант 20	Вариант 21	Вариант 22	Вариант 23	Вариант24	Вариант 25
1	*							*				*							*					*	
2		*							*				*					*					*		
3			*					*		*				*											
4				*					*		*				*										
5	*				*							*				*					*				
6		*				*							*				*					*			
7			*				*							*				*					*		
8				*				*							*				*					*	
9					*				*							*				*	*				*
10	*					*				*							*					*			
11		*					*				*							*					*		
12			*									*				*			*		*			*	
13				*									*				*			*		*			*
14					*					*				*				*					*		
15						*					*				*				*					*	
16					*		*									*				*	*				*
17	*							*				*					*					*			
18		*				*			*				*												
19			*				*			*				*											
20				*							*				*					*					*
21	*							*				*							*					*	
22		*							*				*					*					*		
23			*					*		*				*											
24				*					*		*				*										
25	*				*							*				*					*				
26		*				*							*				*					*			
27			*				*							*				*					*		
28				*				*							*				*					*	
29					*				*							*				*	*				*
30	*					*				*							*					*			
31		*					*				*							*					*		
32			*									*				*			*		*				
33				*									*				*			*		*			*
34					*					*				*				*					*		
35						*					*				*				*					*	
36					*		*									*				*	*				*
37	*							*				*					*					*			
38		*				*			*				*												
39			*				*			*				*											
40				*							*				*						*					*
41	*																			*					
42		*																	*		*				
43			*															*				*			
44				*													*						*		
45					*											*								*	
46						*									*										*
47							*							*											
48								*					*												
49									*			*													
50										*	*														

6.4. ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ ДОЛЖЕН ВКЛЮЧАТЬ

1. Титульный лист по форме с номером варианта.
2. Для каждой задачи из общего списка задач (8 общих задач) и списка задач по вариантам (9 задач по вариантам), всего 17 задач:
    a. Условие задачи.
    b. Блок-схема алгоритма.
    c. Программный код решения этой задачи (листинг).
    d. Скриншоты выполнения программы.

Внимание! Отчет должен быть набран шрифтом **Times New Roman** и отформатирован: поля: левое – 3,5; правое – 1,5; нижнее и верхнее – 2 см; красная строка (отступ) - 1 см; межстрочный интервал – одинарный; правый край выровнен по ширине; рисунки сопровождены надписями.