Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Н.И. Пчелинцева, И.И. Ерохин

Лабораторный практикум по дисциплине

«ПЕРСПЕКТИВНЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»:

учебное пособие

Калуга – 2023

УДК 004.42

Рецензенты:
зав. кафедрой «Информационные системы и сети» (ИУК2)
к.т.н., доцент И.В. Чухраев

руководитель проектного департамента
АО «Калуга-Астрал» Н.И. Мезенцев

Утверждено Методической комиссией КФ МГТУ им. Н.Э. Баумана
(протокол № X от XX.XX.2023 г., рег. номер XXX/МК2023-XX)

Пчелинцева Н.И., Ерохин И.И.

Лабораторный практикум по дисциплине «Перспективные языки программирования»: учебное пособие / Н.И. Пчелинцева, И.И. Ерохин – Калуга: КФ МГТУ им. Н.Э. Баумана, 2023. – 229 с.

В лабораторном практикуме приведены теоретические сведения, примеры и задания для выполнения лабораторных работ по дисциплине «Перспективные языки программирования».

Учебное пособие предназначено для студентов КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Перспективные языки программирования» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Требования к программному обеспечению:

1. Интерпретатор Python версии 3.8 и старше
2. PyCharm или аналог

ЛАБОРАТОРНАЯ РАБОТА №1
РЕАЛИЗАЦИЯ АЛГОРИТМОВ РАЗВЕТВЛЯЮЩЕЙСЯ И ЦИКЛИЧЕСКОЙ СТРУКТУРЫ НА PYTHON

Целью выполнения лабораторной работы является формирование практических навыков процедурного программирования, разработки и отладки программ, овладение методами и средствами разработки и оформления технической документации.

Основными задачами выполнения лабораторной работы являются:

1. Изучить условные конструкции.
2. Изучить циклические конструкции.
3. Ознакомиться со структурой массивов.

Результатами работы являются:

1. Реализация разработанных алгоритмов на языке программирования Python;
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Циклические и разветвляющиеся (условные) алгоритмы связывают условные выражения, которые необходимы для их реализации.

Ряд операций представляют условные выражения. Все эти операции принимают два операнда и возвращают логическое значение, которое в Python представляет тип boolean. Существует только два логических значения - True (выражение истинно) и False (выражение ложно).

Операции сравнения

Простейшие условные выражения представляют операции сравнения, которые сравнивают два значения. Python поддерживает следующие операции сравнения:

1. == (проверка на равенство);
   Возвращает True, если оба операнда равны. Иначе возвращает False.

2. != (проверка на неравенство);
   Возвращает True, если оба операнда НЕ равны. Иначе возвращает False.

3. > (больше чем);
   Возвращает True, если первый операнд больше второго.

4. < (меньше чем);
   Возвращает True, если первый операнд меньше второго.

5. >= (больше или равно);
   Возвращает True, если первый операнд больше или равен второму.

6. <= (меньше или равно);

Возвращает True, если первый операнд меньше или равен второму.

Примеры использования операций сравнения:

```
a = 5
b = 6
result = 5 == 6  # сохраняем результат операции в
переменную
print(result)  # False - 5 не равно 6
print(a != b)  # True
print(a > b)  # False - 5 меньше 6
print(a < b)  # True

bool1 = True
bool2 = False
print(bool1 == bool2)  # False - bool1 не равно bool2
```

УСЛОВНЫЕ КОНСТРУКЦИИ

Условная конструкция if

Условные конструкции используют условные выражения и в зависимости от их значения направляют выполнение программы по одному из путей. Одна из таких конструкций - это конструкция if. Она имеет следующее формальное определение:

```
if логическое_выражение:
    инструкции
[elif логическое выражение:
    инструкции]
[else:
    инструкции]
```

В самом простом виде после ключевого слова if идет логическое выражение. И если это логическое выражение возвращает True, то выполняется последующий блок инструкций, каждая из которых должна начинаться с новой стоки и должна иметь отступы от начала строки:

```
age = 22
if age > 21:
    print("Доступ разрешен")
print("Завершение работы")
```

Результат: Доступ разрешен
Завершение работы

Поскольку в данном случае значение переменной age больше 21, то будет выполняться блок if. Отметим, что отступ для обозначения нового блока желательно делать либо в 4 пробела, либо в кратное 4 число пробелов.

Следует обратить внимание на последнюю строку, которая выводит сообщение "Завершение работы". Она не имеет отступов от начала строки, поэтому она не принадлежит к блоку if и будет выполняться в любом случае, даже если выражение в конструкции if возвратит False.

Но если бы были отступы, то она также принадлежала бы к конструкции if:

```
age = 22
if age > 21:
    print("Доступ разрешен")
    print("Завершение работы")
```

Если вдруг надо определить альтернативное решение на тот случай, если условное выражение возвратит False, то можно использовать блок else:

```
age = 18
if age > 21:
    print("Доступ разрешен")
else:
    print("Доступ запрещен")
```

Если выражение age > 21 возвращает True, то выполняется блок if, иначе выполняется блок else.

Если необходимо ввести несколько альтернативных условий, то можно использовать дополнительные блоки elif, после которого идет блок инструкций.

```
age = 18
if age >= 21:
    print("Доступ разрешен")
elif age >= 18:
    print("Доступ частично разрешен")
else:
    print("Доступ запрещен")
```

Вложенные конструкции if

Конструкция if в свою очередь сама может иметь вложенные конструкции if:

```
age = 18
if age >= 18:
```

```
    print("Больше 17")
    if age > 21:
        print("Больше 21")
    else:
        print("От 18 до 21")
```

Стоит учитывать, что вложенные выражения if также должны начинаться с отступов, а инструкции во вложенных конструкциях также должны иметь отступы. Отступы, расставленные не должным образом, могут изменить логику программы. Так, предыдущий пример НЕ аналогичен следующему:

```
age = 18
if age >= 18:
    print("Больше 17")
if age > 21:
    print("Больше 21")
else:
    print("От 18 до 21")
```

Пример программы, использующей вложенные условные конструкции:

```
#! Программа Обменный пункт
usd = 57
euro = 60
money = int(input("Введите сумму, которую вы хотите
обменять: "))
currency = int(input("Укажите код валюты (доллары -
400, евро - 401): "))
if currency == 400:
    cash = round(money / usd, 2)
    print("Валюта: доллары США")
elif currency == 401:
    cash = round(money / euro, 2)
    print("Валюта: евро")
else:
    cash = 0
    print("Неизвестная валюта")
print("К получению:", cash)
```

С помощью функции input() осуществляется получение вводимых пользователем данных на консоль. Причем данная функция возвращает данные в виде строки, поэтому ее надо привести к целому числу с помощью функции int(), чтобы введенные данные можно было использовать в арифметических операциях.

Программа подразумевает, что пользователь вводит количество средств, которые надо обменять, и код валюты, на которую надо произвести обмен. Коды валюты достаточно условны: 400 для долларов и 401 для евро.

С помощью конструкции if проверяем код валюты и делим на соответствующий валютный курс. Так как в процессе деления образуется довольно длинное число с плавающей точкой, которое может содержать множество знаков после запятой, то оно округляется до двух чисел после запятой с помощью функции round().

В завершении на консоль выводится полученное значение.

Результат работы: Введите сумму, которую вы хотите обменять:
20000
Укажите код валюты (доллары - 400, евро - 401): 401
Валюта: евро
К получению: 333.33

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ

Циклы позволяют повторять некоторое действие в зависимости от соблюдения некоторого условия.

Цикл while

Первый цикл - это цикл while. Он имеет следующее формальное определение:

```
while условное_выражение:
```

инструкции

После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.

Все инструкции, которые относятся к циклу while, располагаются на последующих строках и должны иметь отступ от начала строки.

```
choice = "y"
while choice.lower() == "y":
    print("Привет")
    choice = input("Для продолжения нажмите Y, а для
выхода любую другую клавишу: ")
print("Работа программы завешена")
```

В данном случае цикл while будет продолжаться, пока переменная choice содержит латинскую букву "Y" или "y".

Сам блок цикла состоит из двух инструкций. Сначала выводится сообщение "Привет", а потом вводится новое значение для переменной choice. И если пользователь нажмет какую-то другую клавишу, отличную от Y, произойдет выход из цикла, так как условие choice.lower() == "y" вернет значение False. Каждый такой проход цикла называется итерацией.

Также следует отметить, что последняя инструкция print("Работа программы завешена") не имеет отступов от начала строки, поэтому она не входит в цикл while.

Другой пример - вычисление факториала:

```
#! Программа по вычислению факториала
number = int(input("Введите число: "))
i = 1
factorial = 1
while i <= number:
    factorial *= i
    i += 1
print("Факториал числа", number, "равен", factorial)
```

Здесь вводит с консоли некоторое число, и пока число-счетчик i не будет больше введенного числа, будет выполняться цикл, в котором происходит умножения числа factorial.

Результат работы: Введите число: 6

Факториал числа 6 равен 720

Цикл for

Другой тип циклов представляет конструкция for. Цикл for вызывается для каждого числа в некоторой коллекции чисел. Коллекция чисел создается с помощью функции range(). Формальное определение цикла for:

```
for int_var in функция_range:
    инструкции
```

После ключевого слова for идет переменная int_var, которая хранит целые числа (название переменной может быть любое), затем ключевое слово in, вызов функции range() и двоеточие.

А со следующей строки располагается блок инструкций цикла, которые также должны иметь отступы от начала строки.

При выполнении цикла Python последовательно получает все числа из коллекции, которая создается функцией range, и сохраняет эти числа в переменной int_var. При первом проходе цикл получает первое число из коллекции, при втором - второе число и так далее, пока не переберет все числа. Когда все числа в коллекции будут перебраны, цикл завершает свою работу.

Работа данного цикла на примере вычисления факториала:

```
#! Программа по вычислению факториала
number = int(input("Введите число: "))
factorial = 1
for i in range(1, number+1):
    factorial *= i
print("Факториал числа", number, "равен", factorial)
```

Вначале считывается с консоли число. В цикле определяется переменная i, в которую сохраняются числа из коллекции, создаваемой функцией range.

Функция range здесь принимает два аргумента - начальное число коллекции (здесь число 1) и число, до которого надо добавлять числа (то есть number +1).

Предполагается, с консоли вводится число 6, то вызов функции range приобретает следующую форму:

```
range(1, 6+1):
```

Эта функция будет создавать коллекцию, которая будет начинаться с 1 и будет последовательно наполняться целыми числами вплоть до 7. То есть это будет коллекция [1, 2, 3, 4, 5, 6].

При выполнении цикла из этой коллекции последовательно будут передаваться числа в переменную i, а в самом цикле будет происходить умножение переменной i на переменную factorial. В итоге мы получим факториал числа.

Результат работы: Введите число: 6

Факториал числа 6 равен 720

Функция range

Функция range имеет следующие формы:

7. range(stop): возвращает все целые числа от 0 до stop
8. range(start, stop): возвращает все целые числа в промежутке от start (включая) до stop (не включая). Выше в программе факториала использована именно эта форма.
9. range(start, stop, step): возвращает целые числа в промежутке от start (включая) до stop (не включая), которые увеличиваются на значение step

Примеры вызовов функции range:

```
range(5)              # 0, 1, 2, 3, 4
range(1, 5)           # 1, 2, 3, 4
range(2, 10, 2)       # 2, 4, 6, 8
range(5, 0, -1)       # 5, 4, 3, 2, 1
```

Последовательный вывод чисел от 0 до 4:

```
for i in range(5):
    print(i, end=" ")
```

Вложенные циклы

Одни циклы внутри себя могут содержать другие циклы. Пример вывода таблицы умножения:

```
for i in range(1, 10):
    for j in range(1, 10):
        print(i * j, end="\t")
    print("\n")
```

Внешний цикл for i in range(1, 10) срабатывает 9 раз, так как в коллекции, возвращаемой функцией range, 9 чисел. Внутренний цикл for j in range(1, 10) срабатывает 9 раз для одной итерации внешнего цикла, и соответственно 81 раз для всех итераций внешнего цикла.

В каждой итерации внутреннего цикла на консоль будет выводиться произведение чисел i и j.

Результат работы:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45

6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Выход из цикла. break и continue

Для управления циклом возможно использование специальных операторов break и continue. Оператор break осуществляет выход из цикла. А оператор continue выполняет переход к следующей итерации цикла.

Оператор break может использоваться, если в цикле образуются условия, которые несовместимы с его дальнейшим выполнением.

```
#! Программа Обменный пункт
print("Для выхода нажмите Y")
while True:
    data = input("Введите сумму для обмена: ")
    if data.lower() == "y":
        break  # выход из цикла
    money = int(data)
    cache = round(money / 56, 2)
    print("К выдаче", cache, "долларов")
print("Работа обменного пункта завершена")
```

Здесь бесконечный цикл, так как условие while True всегда истинно и всегда будет выполняться. Это популярный прием для создания программ, которые должны выполняться неопределенно долго.

В самом цикле получаем ввод с консоли. Предполагается, что пользователь будет вводить число - условную сумму денег для обмена. Если пользователь вводит букву "Y" или "y", то с помощью оператора break выходим из цикла и прекращаем работу программы. Иначе делить введенную сумму на обменный курс, с помощью функции round

округлить результат и вывести его на консоль. И так до бесконечности, пока пользователь не захочет выйти из программы, нажав на клавишу Y.

Результат работы: Для выхода нажмите Y
Введите сумму для обмена: 20000
К выдаче 357.14 долларов
Введите сумму для обмена: Y
Работа обменного пункта завершена

Но что, если пользователь введет отрицательное число? В этом случае программа также выдаст отрицательный результат, что не является корректным поведением. И в этом случае перед вычислением можно проверить значение, меньше ли оно нуля, и если меньше, с помощью оператора continue выполнить переход к следующей итерации цикла без его завершения:

```
#! Программа Обменный пункт
print("Для выхода нажмите Y")
while True:
    data = input("Введите сумму для обмена: ")
    if data.lower() == "y":
        break  # выход из цикла
    money = int(data)
    if money < 0:
        print("Сумма должна быть положительной!")
        continue
    cache = round(money / 56, 2)
    print("К выдаче", cache, "долларов")
print("Работа обменного пункта завершена")
```

Для определения, относится ли инструкция к блоку while или к вложенной конструкции if, опять же используются отступы.
И в этом случае нельзя получить результат для отрицательной суммы.

Результат работы: Для выхода нажмите Y
Введите сумму для обмена: -20000
Сумма должна быть положительной!
Введите сумму для обмена: 20000

К выдаче 357.14 долларов
Введите сумму для обмена: y
Работа обменного пункта завершена

МАССИВЫ

Массивом в языке Python называется упорядоченная структура данных, которая используется для хранения однотипных объектов. По своему функциональному назначению они схожи со списками, однако обладают некоторыми ограничениями на тип входных данных, а также их размер. Несмотря на такую особенность, массивы являются достаточно функциональным инструментом по работе с наборами данных в языке программирования Python.

Создание и заполнение

Перед тем как добавить (создать) новый массив в Python 3, необходимо произвести импорт библиотеки, отвечающей за работу с таким объектом. Для этого потребуется добавить строку import array в файл программы. Как уже было сказано ранее, массивы ориентированы на взаимодействие с одним постоянным типом данных, вследствие чего все их ячейки имеют одинаковый размер.

Воспользовавшись функцией array, можно создать новый набор данных. В следующем примере демонстрируется заполнение массива Python — запись целых чисел при помощи метода, предложенного выше.

```
import array
data = array('i', [2, 5, 4, 0, 8])
```

Как можно заметить, функция array принимает два аргумента, первым из которых становится тип создаваемого массива, а на месте второго стоит начальный список его значений. В данном случае i представляет собой целое знаковое число, занимающее 2 байта памяти. Вместо него можно использовать и другие примитивы, такие как 1-

байтовый символ (c) или 4-байтовое число с плавающей точкой (f). При этом важно помнить, что массив способен хранить только данные одного типа, иначе вызов программы завершится ошибкой.

Обратиться к элементу можно при помощи квадратных скобок, к примеру, data[2].

Вывод

При работе с любыми данными в программе время от времени возникает необходимость в их проверке, что можно легко осуществить с помощью вывода на экран. Выполнить подобное действие поможет функция под названием print. Она принимает в качестве аргумента один из элементов созданного и заполненного ранее массива. В следующем примере его обработка производится при помощи цикла for, где каждый элемент массива data получает временный идентификатор i для передачи в упомянутый ранее метод print.

```
import array
data = array('i', [2, 5, 4, 0, 8])
for i in data:
    print(i)
```

Результатом работы приведенного выше кода является вывод массива Python — перебор всех присвоенных ранее целочисленных значений и поочередный вывод в один столбец.

Получение размера

Поскольку размерность массива может меняться во время выполнения программы, иногда бывает полезным узнать текущее количество элементов, входящих в его состав. Функция len служит для получения длины (размера) массива в Python в виде целочисленного значения. Чтобы отобразить в Python количество элементов массива на экране стоит воспользоваться методом print.

```
import array
data = array('i', [2, 5, 4, 0, 8])
print(len(data))
```

Как видно из представленного выше кода, функция print получает в качестве аргумента результат выполнения len, что позволяет ей вывести числовое значение в консоль.

Двумерный массив

В некоторых случаях для правильного представления определенного набора информации обычного одномерного массива оказывается недостаточно. В языке программирования Python 3 двумерных и многомерных массивов не существует, однако базовые возможности этой платформы легко позволяют построить двумерный список. Элементы подобной конструкции располагаются в столбцах и строках, заполняемых как это показано на следующем примере.

```
d1 = []
for j in range(5):
    d2 = []
    for i in range(5):
        d2.append(0)
    d1.append(d2)
```

Здесь можно увидеть, что основная идея реализации двумерного набора данных заключается в создании нескольких списков d2 внутри одного большого списка d1. При помощи двух циклов for происходит автоматическое заполнение нулями матрицы с размерностью 5×5. С этой задачей помогают справляться методы append и range, первый из которых добавляет новый элемент в список (0), а второй позволяет устанавливать его величину (5).

Нельзя не отметить, что для каждого нового цикла for используется собственная временная переменная, выполняющая представление текущего элемента внешнего (j) или внутренних (i) списков. Обратиться к нужной ячейке многомерного списка можно при помощи указания ее координат в квадратных скобках, ориентируясь на строки и столбцы: d1[1][2].

ЗАДАЧИ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Рассмотрим работу циклов с массивами на примере следующей задачи: сформировать прямоугольную матрицу A(10, 20) следующего вида:

$$\begin{pmatrix} 1 & 2 & . & . & . & 20 \\ 1 & 2 & . & . & . & 20 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & . & . & . & 20 \end{pmatrix}$$

Чтобы работать с массивами, необходимо подключить библиотеку array.

```
import array
```

По условию легко посчитать размер матрицы. Нужно задать константы, обозначающие размер:

```
size1=10
size2=20
```

Далее создается массив с помощью функции range():

```
mas=[[[0 for i in range(size1)] for j in
range(size2)]]
```

Затем матрица заполняется соответствующими элементами по закону, заданному в цикле, и выводится на экран:

```
for i in range(size1):
    for j in range(size2):
        mas[i][j]=j+1
        if j<size2-1:
```

```
            print(mas[i][j],"\t")
        else:
            print(mas[i][j])
print()
```

Итоговый код:

```
import array
size1=10
size2=20
mas=[[[0 for i in range(size1)] for j in
range(size2)]]
for i in range(size1):
    for j in range(size2):
        mas[i][j]=j+1
        if j<size2-1:
            print(mas[i][j],"\t")
        else:
            print(mas[i][j])
print()
```

Результат работы: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Необходимо рассмотреть случай, когда массив задан случайным образом в заданном диапазоне. Следующая задача как раз об этом: дан массив A(N). Найти пару соседних элементов, наиболее близко расположенных друг к другу.

Мера близости: $R = |A[i+1] - A[i]|$

Помимо библиотеки array обязательно подключить математическую библиотеку math для вычислений и библиотеку для случайных чисел random:

```
import array
import random
import math
```

Затем выводится приглашение пользователю ввести количество элементов массива и его границу. Далее необходимо сформировать массив:

```
print("Введите количество элементов массива")
n=int(input())
print("Введите границу массива")
upp=int(input())
a=[0 for i in range(n)]
```

Случайным образом формируется массив по закону, описанному в цикле for:

```
for i in range(n):
    a[i]=random.randint(0,upp)
    print("A[",i,"]","=",a[i],"\n")
```

Резервируется переменная index и обнуляется. Она необходима для решения задачи. Формируется цикл с вложенным условием:

```
for i in range(n-1):
    if (abs(a[i+1] - a[i])<abs(a[index+1] -
a[index])):
        index=i
print("Близкорасположенные числа ",a[index],'
',a[index+1])
```

Итоговый код:

```
import array
import random
import math
print("Введите количество элементов массива")
n=int(input())
print("Введите границу массива")
upp=int(input())
a=[0 for i in range(n)]
for i in range(n):
    a[i]=random.randint(0,upp)
    print("A[",i,"]","=",a[i],"\n")
for i in range(n-1):
    if (abs(a[i+1] - a[i])<abs(a[index+1] -
a[index])):
        index=i
print("Близкорасположенные числа ",a[index],'
',a[index+1])
```

Результат работы: Введите количество элементов массива
6
Введите границу массива
7
A[0]=0
A[1]=6
A[2]=2
A[3]=-1
A[4]=-2
A[5]=1
Близкорасположенные числа -1 -2

Рассмотрим одну из наиболее часто встречающихся задач: напечатать таблицу истинности логической функции

$$\left(\overline{A\&B}\right)\vee(A \oplus C),$$

где &, ∨, –, $\oplus$ – знаки логических операций И, ИЛИ, НЕ, неэквивалентность.

Для начала необходимо понять: для какого числа комбинаций необходимо построить таблицу истинности. Их число по законам комбинаторики равно 2^3=8.

Теперь, чтобы построить таблицу эквивалентности, необходимо разделить всю логическую функцию на несколько простых составляющих. Таким образом, вычислим для каждой комбинации переменных конъюнкцию A&B, инверсию данной конъюнкции и сложим результат от второго вычисления с $A \oplus C$.

Таким образом, код программы принимает вид:

```
print("a b c f1 f2 f3")
for a in range(0,2):
    for b in range(0,2):
        for c in range(0,2):
            f1 = a*b
            f2 = 1 - f1
            f3 = f2 +(a^c)
            if (f3==2):
                f3=1
            print("%d"%a,"%d"%b,"%d "%c,"%d "%f1,"%d
"%f2,"%d"%f3)
```

Результат работы: a b c f1 f2 f3

0 0 0 0 1 1
0 0 1 0 1 1
0 1 0 0 1 1
0 1 1 0 1 1
1 0 0 0 1 1
1 0 1 0 1 1
1 1 0 1 0 1
1 1 1 1 0 0

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Вариант 1

Задача 1. Три друга были свидетелями ДТП. Первый заметил, что номер нарушителя делится на 2, 7 и 11. Второй запомнил, что в записи номера участвуют всего две различные цифры, а третий – что сумма цифр равна 30. Определить четырехзначный номер нарушителя.

Задача 2. Сформировать массив IM(100), элементами которого являются числа

2, 1, 4, 3, 6, 5, … , 100, 99.

Задача 3. Дано натуральное число N. Найти сумму цифр числа, находящихся на четных позициях (старшая цифра находится на первой позиции).

Задача 4. Сформировать массив IM(100), элементами которого являются числа

1, 100, 2, 99, 3, 98, … , 50, 51.

Задача 5. Дан целочисленный массив S(26). Сформировать матрицу A, первая строка которой будет содержать элементы массива с четными номерами, а вторая – с нечетными.

Задача 6. Дана целочисленная матрица A(M, N) и натуральное число K < N. Выяснить, все ли элементы K-го столбца матрицы A четные.

Задача 7. Даны вещественные матрица X(15, 20) и массив Y(15). Заменить четные столбцы матрицы X на вектор Y.

Задача 8. Даны вещественные числа A_1, B_1, C_1, A, B, C. Выяснить взаимное расположение прямых $A_1*x + B_1*y = C_1$ и $A*x + B*y = C$. Если прямые пересекаются, напечатать координаты точки пересечения.

Задача 9. Сформировать квадратную матрицу A(15, 15) следующего вида

$$\begin{pmatrix} \mathbf{15} & \mathbf{0} & \mathbf{0} & . & . & \mathbf{0} \\ \mathbf{0} & \mathbf{14} & \mathbf{0} & . & . & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{13} & . & . & \mathbf{0} \\ . & . & . & . & . & . \\ \mathbf{0} & \mathbf{0} & . & . & \mathbf{2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & . & . & \mathbf{1} \end{pmatrix}$$

Задача 10. Зашифровать введенный текст, заменив каждый символ на символ, стоящий через один от данного в таблице кодировки. Исходное разбиение на строки должно быть сохранено.

Вариант 2

Задача 1. Вычислить значения функции

$$f(x) = \sin x + \sin^2 x^2 + \sin^3 x^3$$

для значений аргумента

x от 0.0 до 1.2 с шагом 0.1.

Задача 2. Дано натуральное число *N*. Вычислить сумму *k*-младших (правых) цифр числа.

Задача 3. Три друга были свидетелями ДТП. Первый заметил, что номер нарушителя делится на 2, 7 и 11. Второй запомнил, что в записи номера участвуют всего две различные цифры, а третий – что сумма цифр равна 30. Определить четырехзначный номер нарушителя.

Задача 4. Выяснить, есть ли во введенном тексте слова, начинающиеся с буквы А, и сколько таких слов.

Задача 5. Напечатать таблицу истинности логической функции

$$(\overline{A \vee B}) \& (A \oplus C),$$

где &, ∨, –, ⊕ – знаки логических операций И, ИЛИ, НЕ, Неэквивалентность.

Задача 6. Дан целочисленный массив S(26). Сформировать матрицу A, первая строка которой будет содержать элементы массива с четными номерами, а вторая – с нечетными.

Задача 7. Сформировать целочисленный массив A(75), элементами которого являются случайные числа из диапазона [-5, 40]. Переслать в массив Y все элементы, значения которых меньше 20.

Задача 8. Ввести текст, состоящий только из цифр и букв. Выяснить, верно ли, что сумма числовых значений цифр, находящихся в тексте, равна длине текста.

Задача 9. Сформировать и распечатать квадратную матрицу размерности $M < 20$ следующего вида:

$$\begin{pmatrix} 1 & 0 & 0 & . & . & 0 \\ 2 & 1 & 0 & . & . & 0 \\ 3 & 2 & 1 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ M & M-1 & M-2 & . & . & 1 \end{pmatrix}$$

Задача 10. Имеются N населенных пунктов ($N \leq 15$), и сеть авиалиний, соединяющих эти города. Сеть задана матрицей связности $M(N, N)$, где $M(i, j) = 0$, если города i и j не связаны между собой, и $M(i, j) = 1$ – в противном случае. Выяснить, есть ли среди N населенных пунктов изолированные города (такие, в которые нельзя долететь).

Вариант 3

Задача 1. Сформировать квадратную матрицу A(12, 12) следующего вида

$$\begin{pmatrix} 1 & 2 & 3 & . & . & 12 \\ 0 & 1 & 2 & . & . & 11 \\ 0 & 0 & 1 & . & . & 10 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 \end{pmatrix}$$

Задача 2. Дана целочисленная матрица $A(N, M)$, $(N, M \leq 10)$. Построить по ней целочисленный массив B, присвоив его -му элементу значения 1, если k-я строка матрицы A симметрична (т.е. первый элемент равен последнему, второй - предпоследнему и т.д.), и 0 – в противном случае.

Задача 3. Вычислить значения функции

$$f(x) = \sin x + \sin^2 x^2 + \sin^3 x^3$$

для значений аргумента *x* от 0.0 до 1.2 с шагом 0.1.

Задача 4. Напечатать таблицу истинности логической функции

$$\left(\overline{A \vee B \oplus C}\right) \vee A,$$

где &, ∨, –, $\oplus$ – знаки логических операций И, ИЛИ, НЕ, Неэквивалентность.

Задача 5. Вычислить значения функции

$$f(x) = \begin{cases} \sin\left(\frac{\pi}{8} + |x|\right) & \text{при } x < 0.3, \\ \sin\left(\frac{x^2 \pi}{2}\right) & \text{при } x \geq 0.3 \end{cases}$$

для значений аргумента *x* от -0.5 до 1.2 с шагом 0.1.

Задача 6. Сформировать матрицу B(M, N) элементами которой являются случайные числа, равномерно распределенные в интервале (-5,7). Переставляя её строки и столбцы, добиться того, чтобы наибольший элемент матрицы оказался в правом нижнем углу.

Задача 7. Определить, сколько слов во введенном тексте начинаются и оканчиваются одной и той же буквой. (Слова разделены пробелами.)

Задача 8. Во введенном тексте подсчитать количество слов, содержащих три буквы ‘с’ (слова разделены пробелами).

Задача 9. По введенному символу установить, в каких позициях его двоичного кода записаны единицы.

Задача 10. Дано натуральное число N. Вычислить

$$S = \sum_{k=1}^{N} (-1)^k (2k+1)!$$

Вариант 4

Задача 1. Сформировать прямоугольную матрицу A(10, 20) следующего вида:

$$\begin{pmatrix} 1 & 2 & . & . & . & 20 \\ 1 & 2 & . & . & . & 20 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & . & . & . & 20 \end{pmatrix}$$

Задача 2. Вычислить значения функции

$$f(x,y)=\begin{cases}\ln\left|\dfrac{x}{1+y}\right| & \text{при } x \geq y,\\ \dfrac{1+x}{1+y}e^{-|x+y|} & \text{при } x < y\end{cases}$$

для значений аргумента x от 0.2 до 0.6 с шагом 0.1,
y от 0.0 до 0.4 с шагом 0.05.

Задача 3. Сформировать целочисленный массив IM(N), элементами которого являются случайные числа из диапазона [3…42]. Подсчитать сумму элементов массива, значения которых кратны 8.

Задача 4. Дан массив A(N). Найти пару соседних элементов, наиболее близко расположенных друг к другу. Мера близости:

$$R = |A[i+1] - A[i]|$$

Задача 5. Сформировать массив IM(100), элементами которого являются числа

1, 100, 2, 99, 3, 98, … , 50, 51.

Задача 6. Найти наибольший общий делитель (НОД) двух введенных натуральных чисел, используя алгоритм Евклида.

Алгоритм Евклида: вычитаем из большего числа меньшее до тех пор, пока они не сравняются; полученное в результате число и есть НОД.

Задача 7. В заданном целочисленном массиве распечатать те элементы, порядковые номера которых – числа Фибоначчи.

Задача 8. Даны вещественные числа A_1, B_1, C_1, A, B, C. Выяснить взаимное расположение прямых A_1*x + B_1*y = C_1 и A*x + B*y = C.

Если прямые пересекаются, напечатать координаты точки пересечения.

Задача 9. Вычислить значения функции

$$f(x) = \begin{cases} \ln(1 + |x|) & \text{при } x < -0.2, \\ e^{(-1+x)} & \text{при } x \geq -0.2 \end{cases}$$

для значений аргумента *x* от -0.8 до 0.6 с шагом 0.1.

Задача 10. Сформировать квадратную матрицу A(12, 12) следующего вида

$$\begin{pmatrix} 1 & 2 & 3 & . & . & 12 \\ 0 & 1 & 2 & . & . & 11 \\ 0 & 0 & 1 & . & . & 10 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 \end{pmatrix}$$

Вариант 5

Задача 1. Напечатать таблицу истинности логической функции

$$\left(\overline{A \vee B \& \overline{C}}\right) \vee A,$$

где &, ∨, – – знаки логических операций И, ИЛИ, НЕ.

Задача 2. Дан массив X(178). Вычислить разность между максимальным и минимальным по модулю элементами этого массива.

Задача 3. Сформировать прямоугольную матрицу A(10, 20) следующего вида:

$$\begin{pmatrix} 1 & 2 & . & . & . & 20 \\ 1 & 2 & . & . & . & 20 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & . & . & . & 20 \end{pmatrix}$$

Задача 4. Ввести строку и определить, располагаются ли буквы в ней в порядке, обратном алфавитному.

Задача 5. По заданному вещественному x вычислить значение $\sqrt[3]{x}$ по следующей итерационной формуле:

$$y_{i+1} = \frac{1}{3}(2y_i + \frac{x}{y_i^2}).$$

Начальное приближение: $y_0 = x$.

Итерации прекратить при

$$|y_{i+1} - y_i| < 10^{-5}$$

Задача 6. Дан массив A(N). Найти пару соседних элементов, наиболее близко расположенных друг к другу. Мера близости:

$$R = |A[i+1] - A[i]|$$

Задача 7. Выяснить, есть ли во введенном тексте слова, начинающиеся с буквы А, и сколько таких слов.

Задача 8. Во введенном тексте подсчитать количество слов, считая словом последовательность букв и цифр, начинающуюся с буквы (слова разделены пробелами).

Задача 9. Даны вещественные числа A и B ($A < B$). Найти первый член последовательности

$$a_n = (-1)^n \left(1 + \frac{1}{2} + \cdots + \frac{1}{n}\right), n = 1,2, \ldots,$$

который не принадлежит $[A, B]$.

Задача 10. Ввести текст, состоящий только из цифр и букв. Выяснить, верно ли, что сумма числовых значений цифр, находящихся в тексте, равна длине текста.

Вариант 6

Задача 1. Выяснить, есть ли во введенном тексте слова, оканчивающиеся на ‘f’, и сколько таких слов (слова разделяются пробелами).

Задача 2. В доме N этажей и три лифта. Каждый лифт либо свободен, либо занят. Человек стоит на одном из этажей и должен вызвать ближайший к нему свободный лифт, а если такого нет – то ближайший занятый.

Задача 3. Напечатать таблицу истинности логической функции

$$\left(\overline{A \vee B \& \mathrm{C}}\right) \vee A,$$

где &, ∨, – – знаки логических операций И, ИЛИ, НЕ.

Задача 4. Дано натуральное число N. Вычислить сумму его цифр.

Задача 5. Дан вещественный массив A(N). Отсортировать его таким образом, чтобы все положительные числа находились в начале, а отрицательные - в конце массива и был сохранен исходный порядок следования элементов в обеих группах.

Задача 6. По заданному вещественному x вычислить значение $\sqrt[3]{x}$ по следующей итерационной формуле:

$$y_{i+1} = 0.5 * \left(y_i + 3 * x/(2 * y_i^2 + x/y_i)\right)$$

Начальное приближение:

$$y_0 = x$$

Итерации прекратить при

$$|y_{i+1} - y_i| < 10^{-5}$$

Задача 7. Напечатать таблицу истинности логической функции

$$\left(\overline{A \vee B \oplus C}\right) \vee A,$$

где &, ∨, –, ⊕ – знаки логических операций И, ИЛИ, НЕ, Неэквивалентность.

Задача 8. Дано натуральное число N. Найти сумму цифр числа, находящихся на четных позициях (старшая цифра находится на первой позиции).

Задача 9. Сформировать целочисленный массив A(120), элементами которого являются случайные числа из диапазона [-2…3]. Определить, сколько раз в нем встретились два подряд идущих нулевых элемента.

Задача 10. Во введенном тексте подсчитать количество слов, содержащих три буквы ‘c‘ (слова разделены пробелами).

Вариант 7

Задача 1. Дано натуральное число N. Подсчитать сумму цифр этого числа, находящихся на нечетных позициях (нумерация позиций идет слева направо).

Задача 2. Напечатать таблицу истинности для логической функции

$$(A \Leftrightarrow B\&C) \vee \bar{A},$$

где &, ∨, -, ⇔ – знаки логических операций И, ИЛИ, НЕ, Эквивалентность.

Задача 3. Выяснить, есть ли во введенном тексте слова, оканчивающиеся на ‘f’, и сколько таких слов (слова разделяются пробелами).

Задача 4. Сформировать массив IM(100), элементами которого являются числа

1, -1, 2, -2, … , 50, -50.

Задача 5. Во введенной строке подсчитать наибольшее количество одинаковых букв, идущих подряд.

Задача 6. Дан вещественный массив A(N). Отсортировать его таким образом, чтобы все положительные числа находились в начале, а отрицательные - в конце массива и был сохранен исходный порядок следования элементов в обеих группах.

Задача 7. Сформировать целочисленный массив IM(N), элементами которого являются случайные числа из диапазона [3…42]. Подсчитать сумму элементов массива, значения которых кратны 8.

Задача 8. Три друга были свидетелями ДТП. Первый заметил, что номер нарушителя делится на 2, 7 и 11. Второй запомнил, что в записи номера участвуют всего две различные цифры, а третий - что сумма цифр равна 30. Определить четырехзначный номер нарушителя.

Задача 9. Сформировать квадратную матрицу A(15, 15) следующего вида:

$$\begin{pmatrix} 0 & 0 & . & . & 0 & 1 \\ 0 & 0 & . & . & 1 & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 1 & . & . & 0 & 0 \\ 1 & 0 & . & . & 0 & 0 \end{pmatrix}$$

Задача 10. Найти наибольший общий делитель (НОД) двух введенных натуральных чисел, используя алгоритм Евклида.

Алгоритм Евклида: вычитаем из большего числа меньшее до тех пор, пока они не сравняются; полученное в результате число и есть НОД.

Вариант 8

Задача 1. Даны вещественные числа a и b $(a < b)$. Сформировать матрицу X(10, 10), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке $[a, b]$. Найти

в матрице строку с минимальным элементом и поменять ее местами с первой строкой.

Задача 2. Найти количество трехзначных чисел, кратных 15, но не кратных 30. Распечатать эти числа.

Задача 3. Дано натуральное число N. Подсчитать сумму цифр этого числа, находящихся на нечетных позициях (нумерация позиций идет слева направо).

Задача 4. В кассе имеются только трех- и пятирублевые купюры (это было в далеком 1980 г.). Составить программу, которая “выплачивала “бы такими купюрами любую сумму более 7 рублей.

Задача 5. Элементами массива IM(N) являются числа 0 и 1. Отсортировать этот массив таким образом, чтобы все нули находились в начале, а единицы - в конце массива. Дополнительный массив не заводить.

Задача 6. Во введенной строке подсчитать наибольшее количество одинаковых букв, идущих подряд.

Задача 7. Ввести строку и определить, располагаются ли буквы в ней в порядке, обратном алфавитному.

Задача 8. Вычислить значения функции

$$f(x) = \sin x + \sin^2 x^2 + \sin^3 x^3$$

для значений аргумента *x* от 0.0 до 1.2 с шагом 0.1.

Задача 9. Сформировать массив IM(100), элементами которого являются числа

2, 1, 4, 3, 6, 5, … , 100, 99.

Задача 10. По заданному вещественному x вычислить значение $\sqrt[3]{x}$ по следующей итерационной формуле:

$$y_{i+1} = \frac{1}{3}(2y_i + \frac{x}{y_i^2}).$$

Начальное приближение: $y_0 = x$.

Итерации прекратить при

$$|y_{i+1} - y_i| < 10^{-5}$$

Вариант 9

Задача 1. Напечатать таблицу истинности для логической функции:

$$(A \Leftrightarrow B \Leftrightarrow C) \vee \bar{A},$$

где $\vee$, -, $\Leftrightarrow$ – знаки логических операций ИЛИ, НЕ, Эквивалентность.

Задача 2. По трем введенным вещественным числам выяснить, можно ли построить треугольник с такими длинами сторон, и если можно, то какой это треугольник: равносторонний, равнобедренный, прямоугольный или общего вида.

Задача 3. Даны вещественные числа a и b $(a < b)$. Сформировать матрицу X(10, 10), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке $[a, b]$. Найти в матрице строку с минимальным элементом и поменять ее местами с первой строкой.

Задача 4. Дана квадратная матрица размерности $K < 20$. Найти сумму ее элементов, находящихся на диагонали, “ортогональной” главной.

Задача 5. Проведено измерение роста 70 студентов. Данные записаны в массиве ROST. Разместить в массиве NR номера тех

студентов, чей рост меньше 180см, и подсчитать число таких студентов.

Задача 6. Элементами массива IM(N) являются числа 0 и 1. Отсортировать этот массив таким образом, чтобы все нули находились в начале, а единицы - в конце массива. Дополнительный массив не заводить.

Задача 7. Дано натуральное число N. Вычислить сумму его цифр.

Задача 8. Сформировать квадратную матрицу A(12, 12) следующего вида

$$\begin{pmatrix} 1 & 2 & 3 & . & . & 12 \\ 0 & 1 & 2 & . & . & 11 \\ 0 & 0 & 1 & . & . & 10 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & 1 \end{pmatrix}$$

Задача 9. Дано натуральное число N. Вычислить сумму k-младших (правых) цифр числа.

Задача 10. Дано натуральное число N. Найти сумму цифр числа, находящихся на четных позициях (старшая цифра находится на первой позиции).

Вариант 10

Задача 1. Сформировать две квадратные матрицы одинакового размера по следующим правилам:

$$L_{i,j} = \begin{cases} 2, \text{при } i = j; \\ 0, \text{при } i \neq j; \end{cases} \qquad K_{i,j} = \begin{cases} 7, \text{при } i \geq j; \\ 0, \text{при } i < j; \end{cases}$$

Образовать из них третью матрицу с элементами, равными

$$M_{i,j} = \overline{L_{i,j} + K_{i,j}}$$

Задача 2. Найти и распечатать все натуральные трехзначные числа, равные сумме кубов своих цифр.

Задача 3. Напечатать таблицу истинности для логической функции:

$$(A \Leftrightarrow B \Leftrightarrow C) \vee \bar{A},$$

где $\vee$, -, $\Leftrightarrow$ – знаки логических операций ИЛИ, НЕ, Эквивалентность.

Задача 4. Определить k-ю цифру последовательности

182764125216343 … ,

в которой выписаны подряд кубы натуральных чисел.

Задача 5. Определить, является ли введенная строка правильной записью целого шестнадцатеричного числа без знака.

Задача 6. Проведено измерение роста 70 студентов. Данные записаны в массиве ROST. Разместить в массиве NR номера тех студентов, чей рост меньше 180 см, и подсчитать число таких студентов.

Задача 7. Сформировать массив IM(100), элементами которого являются числа

1 , -1 , 2 , -2 , … , 50 , -50.

Задача 8. Сформировать прямоугольную матрицу A(10, 20) следующего вида:

$$\begin{pmatrix} 1 & 2 & . & . & . & 20 \\ 1 & 2 & . & . & . & 20 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & . & . & . & 20 \end{pmatrix}$$

Задача 9. Дана целочисленная матрица A(N, M), $(N, M \leq 10)$. Построить по ней целочисленный массив B, присвоив его k-у элементу значения 1, если k-я строка матрицы A симметрична (т.е. первый элемент равен последнему, второй - предпоследнему и т.д.), и 0 - в противном случае.

Задача 10. Три друга были свидетелями ДТП. Первый заметил, что номер нарушителя делится на 2, 7 и 11. Второй запомнил, что в записи номера участвуют всего две различные цифры, а третий - что сумма цифр равна 30. Определить четырехзначный номер нарушителя.

Вариант 11

Задача 1. В выражении ((((1 ? 2) ? 3) ? 4) ? 5) ? 6 вместо каждого знака ? поставить знак одной из операций +, -, *, / так, чтобы результат вычислений был равен 35.

Задача 2. Дана матрица B(N, M) ($N < M, M > 13$). Найти сумму элементов каждого столбца матрицы. Сформировать массив D из найденных сумм.

Задача 3. Сформировать две квадратные матрицы одинакового размера по следующим правилам:

$$L_{i,j} = \begin{cases} 2, \text{при } i = j; \\ 0, \text{при } i \neq j; \end{cases} \qquad K_{i,j} = \begin{cases} 7, \text{при } i \geq j; \\ 0, \text{при } i < j; \end{cases}$$

Образовать из них третью матрицу с элементами, равными

$$M_{i,j} = \overline{L_{i,j} + K_{i,j}}$$

Задача 4. Вычислить значения функции:

$$f(x) = \begin{cases} \sin\left(\frac{\pi}{2}\right) \text{ при } x \le 0.5; \\ \sin\left((x-1) * \frac{\pi}{2}\right) \text{ при } x > 0.5; \end{cases}$$

для значений аргумента x от –0.4 до 1.3 с шагом 0.1.

Задача 5. Распечатать введенную строку, заменив строчные буквы прописными и повторив дважды каждую цифру.

Задача 6. Определить, является ли введенная строка правильной записью целого шестнадцатеричного числа без знака.

Задача 7. В кассе имеются только трех- и пятирублевые купюры (это было в далеком 1980 г.). Составить программу, которая “выплачивала“ бы такими купюрами любую сумму более 7 рублей.

Задача 8. Напечатать таблицу истинности логической функции

$$(\overline{A \vee B \,\&\, C}) \vee \text{A},$$

где &, $\vee$, - - знаки логических операций И, ИЛИ, НЕ.

Задача 9. Вычислить значения функции

$$f(x,y) = \begin{cases} \ln\left|\frac{x}{1+y}\right| \text{ при } x \ge y, \\ \frac{1+x}{1+y} e^{-x+y} \text{ при } x < y \end{cases}$$

для значений аргументов

x от 0.2 до 0.6 с шагом 0.1;
y от 0.0 до 0.4 с шагом 0.05.

Задача 10. Вычислить значения функции

$$f(x) = \sin x + \sin^2 x^2 + \sin^3 x^3$$

для значений аргумента

x от 0.0 до 1.2 с шагом 0.1.

Вариант 12

Задача 1. Дано натуральное число N. Вычислить

$$S = \sum_{k=1}^{N} \ln\left|\frac{k}{1+k}\right| + k^2$$

Задача 2. Целое число M задано массивом своих двоичных цифр. Напечатать массив двоичных цифр числа M + 1.

Задача 3. В выражении ((((1 ? 2) ? 3) ? 4) ? 5) ? 6 вместо каждого знака ? поставить знак одной из операций +, -, *, / так, чтобы результат вычислений был равен 35.

Задача 4. Треугольник задан координатами своих вершин. Найти его периметр и площадь. Для нахождения длины стороны треугольника использовать директиву #define.

Задача 5. По введенному символу установить, в каких позициях его двойного кода записаны нули.

Задача 6. Распечатать введенную строку, заменив строчные буквы прописными и повторив дважды каждую цифру.

Задача 7. Дана квадратная матрица размерности K < 20. Найти сумму ее элементов, находящихся на диагонали, “ортогональной” главной.

Задача 8. Выяснить, есть ли во введенном тексте слова, оканчивающиеся на ‘f ’, и сколько таких слов. Слова разделяются пробелами.

Задача 9. Дан массив X(178). Вычислить разность между максимальным и минимальным по модулю элементами этого массива.

Задача 10. Сформировать квадратную матрицу A(12, 12) следующего вида

$$\begin{pmatrix} 1 & 2 & 3 & . & . & 12 \\ 0 & 1 & 2 & . & . & 11 \\ 0 & 0 & 1 & . & . & 10 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & 1 \end{pmatrix}$$

Вариант 13

Задача 1. Найти все трехзначные числа, которые можно представить разностью между квадратом числа, образованного первыми двумя цифрами, и квадратом третьей цифры.

Задача 2. Сформировать матрицу B(M, N) элементами которой являются случайные числа, равномерно распределенные в интервале (-5, 7). Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент матрицы оказался в правом нижнем углу.

Задача 3. Дано натуральное число N. Вычислить

$$S = \sum_{k=1}^{N} \ln\left|\frac{k}{1+k}\right| + k^2$$

Задача 4. Дан целочисленный массив A(M). Определить, образуют ли элементы этого массива неубывающую последовательность.

Задача 5. Назовем шестизначный автобусный билет удачным, если сумма его цифр делится на 7. Могут ли два билета подряд быть удачными?

Задача 6. По введенному символу установить, в каких позициях его двойного кода записаны нули.

Задача 7. Определить k-ю цифру последовательности

182764125216343 … ,

в которой выписаны подряд кубы натуральных чисел.

Задача 8. Дано натуральное число N. Подсчитать сумму цифр этого числа, находящихся на нечетных позициях (нумерация позиций идет слева направо).

Задача 9. В доме N этажей и три лифта. Каждый лифт либо свободен, либо занят. Человек стоит на одном из этажей и должен вызвать ближайший к нему свободный лифт, а если такого нет – то ближайший занятый.

Задача 10. Сформировать прямоугольную матрицу A(10, 20) следующего вида:

$$\begin{pmatrix} 1 & 2 & . & . & . & 20 \\ 1 & 2 & . & . & . & 20 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & . & . & . & 20 \end{pmatrix}$$

Вариант 14

Задача 1. Дан массив X(100). Переписать в массив Y элементы массива X с нечетными номерами, а в массив Z - элементы массива X, значения которых кратны пяти.

Задача 2. Определить k-ю цифру последовательности

1 4 9 1 6 2 5 3 6 4 9 … ,

в которой выписаны подряд квадраты всех натуральных чисел.

Задача 3. Найти все трехзначные числа, которые можно представить разностью между квадратом числа, образованного первыми двумя цифрами, и квадратом третьей цифры.

Задача 4. В течении суток через каждый час проведены 24 замера напряжения в сети. Определить максимальное значения напряжения в сети в интервале (20, 6) час и время, когда оно было зафиксировано.

Задача 5. Сформировать матрицу C(N, M), элементами которой являются случайные числа, равномерно распределенные в интервале (-4.0, 8.2). Переставляя ее строки и столбцы, добиться того, чтобы наименьший элемент этой матрицы оказался в левом верхнем углу.

Задача 6. Назовем шестизначный автобусный билет удачным, если сумма его цифр делится на 7. Могут ли два билета подряд быть удачными?

Задача 7. Вычислить значения функции:

$$f(x) = \begin{cases} \sin\left(\frac{\pi}{2}\right) \text{ при } x \le 0.5; \\ \sin\left((x-1) * \frac{\pi}{2}\right) \text{ при } x > 0.5; \end{cases}$$

для значений аргумента x от –0.4 до 1.3 с шагом 0.1.

Задача 8. Даны вещественные числа a и b (a < b). Сформировать матрицу X(10, 10), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b]. Найти в матрице строку с минимальным элементом и поменять ее местами с первой строкой.

Задача 9. Напечатать таблицу истинности для логической функции

$$(A \Leftrightarrow B \,\&\, C) \vee \overline{A},$$

где &, $\vee$, -, $\Leftrightarrow$ - знаки логических операций И, ИЛИ, НЕ, Эквивалентность.

Задача 10. Напечатать таблицу истинности логической функции

$$(\overline{A \vee B \,\&\, C}) \vee A,$$

где &, $\vee$, - - знаки логических операций И, ИЛИ, НЕ.

Вариант 15

Задача 1. Даны вещественные числа a и b (a < b). Сформировать матрицу X(10, 10), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b]. Определить сумму элементов матрицы, сумма номеров строк и столбцов которых равна n.

Задача 2. В заданном целочисленном массиве определить количество перемен знаков.

Задача 3. Дан массив X(100). Переписать в массив Y элементы массива X с нечетными номерами, а в массив Z - элементы массива X, значения которых кратны пяти.

Задача 4. Подсчитать, сколько раз во введенном тексте встречается слово “no“. Слова в тексте разделяются пробелами.

Задача 5. Сформировать целочисленный массив A(85), элементами которого являются случайные числа из диапазона [-20…10]. Найти величину наибольшего среди отрицательных чисел этого массива.

Задача 6. Сформировать матрицу C(N, M), элементами которой являются случайные числа, равномерно распределенные в интервале (-

4.0, 8.2). Переставляя ее строки и столбцы, добиться того, чтобы наименьший элемент этой матрицы оказался в левом верхнем углу.

Задача 7. Треугольник задан координатами своих вершин. Найти его периметр и площадь.

Для нахождения длины стороны треугольника использовать директиву #define.

Задача 8. Напечатать таблицу истинности для логической функции:

$$(A \Leftrightarrow B \Leftrightarrow C) \vee \bar{A},$$

где $\vee$, -, $\Leftrightarrow$ - знаки логических операций ИЛИ, НЕ, Эквивалентность.

Задача 9. Найти количество трехзначных чисел, кратных 15, но не кратных 30. Распечатать эти числа.

Задача 10. Выяснить, есть ли во введенном тексте слова, оканчивающиеся на ‘f’, и сколько таких слов. Слова разделяются пробелами.

Вариант 16

Задача 1. Даны вещественные числа a и b (a < b). Сформировать матрицу P(15, 20), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b]. Найти в матрице два наименьших по модулю элемента.

Задача 2. Дана целочисленная матрица A(M, N) и натуральное число K < N. Выяснить. все ли элементы K-го столбца матрицы A четные.

Задача 3. Даны вещественные числа a и b (a < b). Сформировать матрицу X(10, 10), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b].

Определить сумму элементов матрицы, сумма номеров строк и столбцов которых равна n.

Задача 4. В массиве Y(150) найти наименьший из положительных элементов.

Задача 5. Введите свой год, месяц и день рождения. Ваш день рождения очень счастливый, просто счастливый или обычный?

Очень счастливый - если все остатки от деления на 7 сумм цифр года, месяца и дня совпадают. Просто счастливый - если совпадают два любых остатка. Обычный - если совпадений нет.

Задача 6. Сформировать целочисленный массив A(85), элементами которого являются случайные числа из диапазона [-20…10]. Найти величину наибольшего среди отрицательных чисел этого массива.

Задача 7. Дан целочисленный массив A(M). Определить, образуют ли элементы этого массива неубывающую последовательность.

Задача 8. Сформировать две квадратные матрицы одинакового размера по следующим правилам:

$$L_{i,j} = \begin{cases} 2, \text{при } i = j; \\ 0, \text{при } i \neq j; \end{cases} \qquad K_{i,j} = \begin{cases} 7, \text{при } i \geq j; \\ 0, \text{при } i < j; \end{cases}$$

Образовать из них третью матрицу с элементами, равными

$$M_{i,j} = \overline{L_{i,j} + K_{i,j}}$$

Задача 9. По трем введенным вещественным числам выяснить, можно ли построить треугольник с такими длинами сторон, и если можно, то какой это треугольник: равносторонний, равнобедренный, прямоугольный или общего вида.

Задача 10. Дано натуральное число N. Подсчитать сумму цифр этого числа, находящихся на нечетных позициях (нумерация позиций идет слева направо).

Вариант 17

Задача 1. Во введенном тексте подсчитать количество слов, считая словом последовательность букв и цифр, начинающуюся с буквы. Слова разделены пробелами.

Задача 2. Дан целочисленный массив S(34). Сформировать матрицу A, первая строка которой будет содержать элементы массива с четными номерами, а вторая – с нечетными.

Задача 3. Даны вещественные числа a и b (a < b). Сформировать матрицу P(15, 20), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b]. Найти в матрице два наименьших по модулю элемента.

Задача 4. В магазине стоит очередь из N человек. Время обслуживания i-го покупателя t_i - случайная величина, распределенная по закону равномерной плотности в интервале [2.5, 10.4].

Получить C_1, C_2, … , C_n - время пребывания в очереди каждого покупателя. Указать номер того человека, для обслуживания которого потребовалось минимальное время.

Задача 5. Даны вещественные матрица X(15, 20) и массив Y(15). Заменить четные столбцы матрицы на вектор Y.

Задача 6. Введите свой год, месяц и день рождения. Ваш день рождения очень счастливый, просто счастливый или обычный?

Очень счастливый - если все остатки от деления на 7 сумм цифр года, месяца и дня совпадают. Просто счастливый - если совпадают два любых остатка. Обычный - если совпадений нет.

Задача 7. В течении суток через каждый час проведены 24 замера напряжения в сети. Определить максимальное значения напряжения в сети в интервале (20, 6) час и время, когда оно было зафиксировано.

Задача 8. В выражении ((((1 ? 2) ? 3) ? 4) ? 5) ? 6 вместо каждого знака ? поставить знак одной из операций +, -, *, / так, чтобы результат вычислений был равен 35.

Задача 9. Найти и распечатать все натуральные трехзначные числа, равные сумме кубов своих цифр.

Задача 10. Даны вещественные числа a и b (a < b). Сформировать матрицу X(10, 10), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b]. Найти в матрице строку с минимальным элементом и поменять ее местами с первой строкой.

Вариант 18

Задача 1. Натуральное число m представить в виде суммы квадратов двух натуральных чисел. Выдать сообщение, если такое представление невозможно.

Задача 2. Напечатать таблицу истинности логической функции

$$(\overline{A \vee B}) \,\&\, (A \oplus C),$$

где &, $\vee$, -, $\oplus$ - знаки логических операций И, ИЛИ, НЕ, Неэквивалентность.

Задача 3. Во введенном тексте подсчитать количество слов, считая словом последовательность букв и цифр, начинающуюся с буквы. Слова разделены пробелами.

Задача 4. Сформировать вещественный массив A1(75), элементами которого являются случайные числа из диапазона [16…53]. Переслать из него в массив A2 все элементы, значения которых больше 25.8 и меньше 34.7.

Задача 5. Сформировать целочисленный массив A(75), элементами которого являются случайные числа из диапазона [-5, 40]. Переслать в массив Y все элементы, значения которых меньше 20.

Задача 6. Даны вещественные матрица X(15, 20) и массив Y(15). Заменить четные столбцы матрицы на вектор Y.

Задача 7. Подсчитать, сколько раз во введенном тексте встречается слово “no“. Слова в тексте разделяются пробелами.

Задача 8. Дано натуральное число N. Вычислить

$$S = \sum_{k=1}^{N} \ln\left|\frac{k}{1+k}\right| + k^2$$

Задача 9. Дана матрица B(N, M) ($N < M, M > 13$). Найти сумму элементов каждого столбца матрицы. Сформировать массив D из найденных сумм.

Задача 10. Напечатать таблицу истинности для логической функции:

$$(A \Leftrightarrow B \Leftrightarrow C) \vee \bar{A},$$

где $\vee$, -, $\Leftrightarrow$ - знаки логических операций ИЛИ, НЕ, Эквивалентность.

Вариант 19

Задача 1. Сформировать квадратную матрицу (15, 15) следующего вида:

$$\begin{pmatrix} 1 & 1 & 1 & . & . & 1 \\ 1 & 2 & 2 & . & . & 2 \\ 1 & 2 & 3 & . & . & 3 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & 3 & . & . & 15 \end{pmatrix}$$

Задача 2. Вычислить значения функции

$$f(x) = \begin{cases} \ln(1 + |x|) & \text{при } x < -0.2, \\ e^{(-1+x)} & \text{при } x \geq -0.2 \end{cases}$$

для значений аргумента *x* от -0.8 до 0.6 с шагом 0.1.

Задача 3. Натуральное число m представить в виде суммы квадратов двух натуральных чисел. Выдать сообщение, если такое представление невозможно.

Задача 4. Даны вещественные числа a и b (a < b). Сформировать матрицу X(17, 20), элементами которой являются вещественные случайные числа, равномерно распределенные на отрезке [a, b]. Определить сумму элементов, номера строк которых кратны 3, а столбцов 4.

Задача 5. Определить, сколько слов во введенном тексте начинаются и оканчиваются одной и той же буквой. Слова разделены пробелами.

Задача 6. В массиве Y(150) найти наименьший из положительных элементов.

Задача 7. Сформировать целочисленный массив A(75), элементами которого являются случайные числа из диапазона [-5, 40]. Переслать в массив Y все элементы, значения которых меньше 20.

Задача 8. Найти все трехзначные числа, которые можно представить разностью между квадратом числа, образованного первыми двумя цифрами, и квадратом третьей цифры.

Задача 9. Целое число M задано массивом своих двоичных цифр. Напечатать массив двоичных цифр числа M + 1.

Задача 10. Сформировать две квадратные матрицы одинакового размера по следующим правилам:

$$L_{i,j} = \begin{cases} 2, \text{при } i = j; \\ 0, \text{при } i \neq j; \end{cases} \qquad K_{i,j} = \begin{cases} 7, \text{при } i \geq j; \\ 0, \text{при } i < j; \end{cases}$$

Образовать из них третью матрицу с элементами, равными

$$M_{i,j} = \overline{L_{i,j} + K_{i,j}}$$

Вариант 20

Задача 1. Выяснить, какие цифры (по одной справа и слева) надо приписать к числу 1022, чтобы полученное число делилось на 7, 8, 9.

Задача 2. Дан массив A(N). Найти пару соседних элементов, наиболее близко расположенных друг к другу. Мера близости:

R = | A [i + 1] - A [i] |

Задача 3. Сформировать квадратную матрицу (15, 15) следующего вида:

$$\begin{pmatrix} 1 & 1 & 1 & . & . & 1 \\ 1 & 2 & 2 & . & . & 2 \\ 1 & 2 & 3 & . & . & 3 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & 2 & 3 & . & . & 15 \end{pmatrix}$$

Задача 4. Результаты сдачи экзамена группой из N студентов находятся в массиве REZ. Подсчитать количество студентов, сдавших экзамен на "хорошо" и "отлично"

Задача 5. Сформировать массив IM(100), элементами которого являются числа

1, 100, 2, 99, 3, 98, … , 50, 51.

Задача 6. Определить, сколько слов во введенном тексте начинаются и оканчиваются одной и той же буквой. Слова разделены пробелами.

Задача 7. В магазине стоит очередь из N человек. Время обслуживания i-го покупателя t_i - случайная величина, распределенная по закону равномерной плотности в интервале [2.5, 10.4].

Получить C_1, C_2, … , C_n - время пребывания в очереди каждого покупателя. Указать номер того человека, для обслуживания которого потребовалось минимальное время.

Задача 8. Дан массив X(100). Переписать в массив Y элементы массива X с нечетными номерами, а в массив Z - элементы массива X, значения которых кратны пяти.

Задача 9. Сформировать матрицу B(M, N) элементами которой являются случайные числа, равномерно распределенные в интервале (-

5, 7). Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент матрицы оказался в правом нижнем углу.

Задача 10. В выражении ((((1 ? 2) ? 3) ? 4) ? 5) ? 6 вместо каждого знака ? поставить знак одной из операций +, -, *, / так, чтобы результат вычислений был равен 34.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка заданий.
4. Блок-схемы для каждого задания.
5. Листинг программы для каждого задания.
6. Результаты выполнения программ.
7. Выводы по работе в целом.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Что связывает циклические и условные алгоритмы?
2. Расскажите об условных выражениях.
3. Напишите программу, используя вложенные условия: если число больше 100, то вывести его на экран, если больше 100, но меньше 170, то вывести сообщение-приветствие, иначе вывести сообщение об ошибке.
4. Объясните работу цикла while.
5. Можно ли записать цикл for без использования range()?
6. Как создать массив, состоящий из 10 чисел в промежутке [5..9]?
7. На примере задачи расскажите о заполнении массива случайными числами.
8. Зачем нужна библиотека array?

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Разработать программы согласно варианту.
4. Выполнить тестирование программ.
5. Продемонстрировать работу программ преподавателю.
6. Оформить отчет.
7. Защитить выполненную работу у преподавателя.

ЛАБОРАТОРНАЯ РАБОТА №2
РАБОТА С ФАЙЛАМИ И СТРОКАМИ НА PYTHON

Целью выполнения лабораторной работы является приобретение практических навыков, необходимых для разработки задач, решение которых предполагает использование файлов и строк средствами языка Python.

Основными задачами выполнения домашней работы являются:
Задачи:

4. Изучить основные методы работы с файлами;
5. Изучить основные методы работы со строками;
6. Изучить типовые алгоритмы решения задач с использованием дополнительных библиотек.

Результатами работы являются:

3. Реализация разработанных алгоритмов на языке программирования Python;
4. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Сложные структуры данных в Python

Для работы с наборами данных Python предоставляет такие встроенные типы как списки, кортежи и словари.

Список

Список (list) представляет тип данных, который хранит набор или последовательность элементов. Для создания списка в квадратных скобках ([]) через запятую перечисляются все его элементы. Во многих языках программирования есть аналогичная структура данных, которая называется массив. Например, определим список чисел:

```
numbers = [1, 2, 3, 4, 5]
```

Также для создания списка можно использовать конструктор list():

```
numbers1 = []
numbers2 = list()
```

Оба этих определения списка аналогичны - они создают пустой список.Конструктор list для создания списока может принимать другой список:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(numbers)
```

Для обращения к элементам списка надо использовать индексы, которые представляют номер элемента в списка. Индексы начинаются с нуля. То есть второй элемент будет иметь индекс 1. Для обращения к элементам с конца можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего - -2 и так далее.

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])    # 1
```

```
print(numbers[2])    # 3
print(numbers[-3])   # 3

numbers[0] = 125  # изменяем первый элемент списка
print(numbers[0])    # 125
```

Если необходимо создать список, в котором повторяется одно и то же значение несколько раз, то можно использовать символ звездочки *. Например, определим список из шести пятерок:

```
numbers = [5] * 6  # [5, 5, 5, 5, 5, 5]
print(numbers)
```

Кроме того, если нам необходим последовательный список чисел, то для его создания удобно использовать функцию range, которая имеет три формы:

1. range(end) - создается набор чисел от 0 до числа end;
2. range(start, end) - создается набор чисел от числа start до числа end;
3. range(start, end, step): создается набор чисел от числа start до числа end с шагом step.

```
numbers = list(range(10))
print(numbers)       # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers = list(range(2, 10))
print(numbers)       # [2, 3, 4, 5, 6, 7, 8, 9]
numbers = list(range(10, 2, -2))
print(numbers)       # [10, 8, 6, 4]
```

Например, следующие два определения списка будут аналогичны, но за счет функции range мы сокращаем объем кода:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(range(1, 10))
```

Список необязательно должен содержать только однотипные объекты. Мы можем поместить в один и тот же список одновременно строки, числа, объекты других типов данных:

```
objects = [1, 2.6, "Hello", True]
```

Перебор элементов

Для перебора элементов можно использовать как цикл for, так и цикл while.

Перебор с помощью цикла for:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]
for item in companies:
    print(item)
```

Здесь вместо функции range мы сразу можем подставить имеющийся список companies.

Перебор с помощью цикла while:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]
i = 0
while i < len(companies):
    print(companies[i])
    i += 1
```

Для перебора с помощью функции len() получаем длину списка. С помощью счетчика i выводит по элементу, пока значение счетчика не станет равно длине списка.

Сравнение списков

Два списка считаются равными, если они содержат один и тот же набор элементов:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(range(1,10))
if numbers == numbers2:
    print("numbers equal to numbers2")
else:
    print("numbers is not equal to numbers2")
```

В данном случае оба списка будут равны.

Методы и функции по работе со списками

Для управления элементами списки имеют целый ряд методов. Некоторые из них:

1. append(item): добавляет элемент item в конец списка;
2. insert(index, item): добавляет элемент item в список по индексу index;
3. remove(item): удаляет элемент item. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение ValueError;
4. clear(): удаление всех элементов из списка;
5. index(item): возвращает индекс элемента item. Если элемент не найден, генерирует исключение ValueError;
6. pop([index]): удаляет и возвращает элемент по индексу index. Если индекс не передан, то просто удаляет последний элемент;
7. count(item): возвращает количество вхождений элемента item в список;
8. sort([key]): сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра key мы можем передать функцию сортировки;
9. reverse(): расставляет все элементы в списке в обратном порядке.

Кроме того, Python предоставляет ряд встроенных функций для работы со списками:

1. len(list): возвращает длину списка;
2. sorted(list, [key]): возвращает отсортированный список;
3. min(list): возвращает наименьший элемент списка;
4. max(list): возвращает наибольший элемент списка.

Добавление и удаление элементов

Для добавления элемента применяются методы append() и insert, а для удаления - методы remove(), pop() и clear().

Использование методов:

```
users = ["Tom", "Bob"]

# добавляем в конец списка
users.append("Alice")  # ["Tom", "Bob", "Alice"]
# добавляем на вторую позицию
users.insert(1, "Bill")    # ["Tom", "Bill", "Bob",
"Alice"]

# получаем индекс элемента
i = users.index("Tom")
# удаляем по этому индексу
removed_item = users.pop(i)      # ["Bill", "Bob",
"Alice"]

last_user = users[-1]
# удаляем последний элемент
users.remove(last_user)  # ["Bill", "Bob"]

print(users)

# удаляем все элементы
users.clear()
```

Проверка наличия элемента

Если определенный элемент не найден, то методы remove и index генерируют исключение. Чтобы избежать подобной ситуации, перед операцией с элементом можно проверять его наличие с помощью ключевого слова in:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]
item = "Oracle"  # элемент для удаления
if item in companies:
    companies.remove(item)

print(companies)
```

Выражение item in companies возвращает True, если элемент item имеется в списке companies. Поэтому конструкция if item in companies

может выполнить последующий блок инструкций в зависимости от наличия элемента в списке.

Подсчет вхождений

Если необходимо узнать, сколько раз в списке присутствует тот или иной элемент, то можно применить метод count():

```
users = ["Tom", "Bob", "Alice", "Tom", "Bill", "Tom"]

users_count = users.count("Tom")
print(users_count)      # 3
```

Сортировка

Для сортировки по возрастанию применяется метод sort():

```
users = ["Tom", "Bob", "Alice", "Sam", "Bill"]

users.sort()
print(users)        # ["Alice", "Bill", "Bob", "Sam",
"Tom"]
```

Если необходимо отсортировать данные в обратном порядке, то мы можем после сортировки применить метод reverse():

```
users = ["Tom", "Bob", "Alice", "Sam", "Bill"]

users.sort()
users.reverse()
print(users)         # ["Tom", "Sam", "Bob", "Bill",
"Alice"
```

При сортировке фактически сравниваются два объекта, и который из них "меньше", ставится перед тем, который "больше". Понятия "больше" и "меньше" довольно условны. И если для чисел все просто - числа расставляются в порядке возрастания, то для строк и других объектов ситуация сложнее. В частности, строки оцениваются по первым символам. Если первые символы равны, оцениваются вторые символы и так далее. При чем цифровой символ считается "меньше", чем алфавитный заглавный символ, а заглавный символ считается

меньше, чем строчный. Подробнее про сравнение строк описывалось в статье Операции со строками.

Таким образом, если в списке сочетаются строки с верхним и нижним регистром, то мы можем получить не совсем корректные результаты, так как для нас строка "bob" должна стоять до строки "Tom". И чтобы изменить стандартное поведение сортировки, мы можем передать в метод sort() в качестве параметра функцию:

```
users = ["Tom", "bob", "alice", "Sam", "Bill"]

users.sort(key=str.lower)
print(users) # ["alice", "Bill", "bob", "Sam", "Tom"]
```

Кроме метода sort мы можем использовать встроенную функцию sorted, которая имеет две формы:

1. sorted(list) - сортирует список list;
2. sorted(list, key) - сортирует список list, применяя к элементам функцию key.

```
users = ["Tom", "bob", "alice", "Sam", "Bill"]

sorted_users = sorted(users, key=str.lower)
print(sorted_users)# ["alice", "Bill", "bob", "Sam",
"Tom"]
```

При использовании этой функции следует учитывать, что эта функция не изменяет сортируемый список, а все отсортированные элементы она помещает в новый список, который возвращается в качестве результата.

Минимальное и максимальное значения

Встроенный функции Python min() и max() позволяют найти минимальное и максимальное значения соответственно:

```
numbers = [9, 21, 12, 1, 3, 15, 18]
print(min(numbers))      # 1
print(max(numbers))      # 21
```

Копирование списков

При копировании списков следует учитывать, что списки представляют изменяемый (mutable) тип, поэтому если обе переменных будут указывать на один и тот же список, то изменение одной переменной, затронет и другую переменную:

```
users1 = ["Tom", "Bob", "Alice"]
users2 = users1
users2.append("Sam")
# users1 и users2 указывают на один и тот же список
print(users1)    # ["Tom", "Bob", "Alice", "Sam"]
print(users2)    # ["Tom", "Bob", "Alice", "Sam"]
```

Это так называемое "поверхностное копирование" (shallow copy). И, как правило, такое поведение нежелательное. И чтобы происходило копирование элементов, но при этом переменные указывали на разные списки, необходимо выполнить глубокое копирование (deep copy). Для этого можно использовать метод deepcopy(), который определен во встроенном модуле copy:

```
import copy
users1 = ["Tom", "Bob", "Alice"]
users2 = copy.deepcopy(users1)
users2.append("Sam")
# пееменные users1 и users2 указывают на разные списки
print(users1)    # ["Tom", "Bob", "Alice"]
print(users2)    # ["Tom", "Bob", "Alice", "Sam"]
```

Копирование части списка

Если необходимо скопировать не весь список, а только его какую-то определенную часть, то мы можем применять специальный синтаксис. который может принимать следующие формы:

1. list[:end]: через параметр end передается индекс элемента, до которого нужно копировать список;

2. list[start:end]: параметр start указывает на индекс элемента, начиная с которого надо скопировать элементы;
3. list[start:end:step]: параметр step указывает на шаг, через который будут копироваться элементы из списка. По умолчанию этот параметр равен 1.

```
users = ["Tom", "Bob", "Alice", "Sam", "Tim", "Bill"]

slice_users1 = users[:3]    # с 0 по 3
print(slice_users1)    # ["Tom", "Bob", "Alice"]

slice_users2 = users[1:3]    # с 1 по 3
print(slice_users2)    # ["Bob", "Alice"]

slice_users3 = users[1:6:2]    # с 1 по 6 с шагом 2
print(slice_users3)    # ["Bob", "Sam", "Bill"]
```

Соединение списков

Для объединения списков применяется операция сложения (+):

```
users1 = ["Tom", "Bob", "Alice"]
users2 = ["Tom", "Sam", "Tim", "Bill"]
users3 = users1 + users2
print(users3)     # ["Tom", "Bob", "Alice", "Tom", "Sam", "Tim", "Bill"]
```

Списки списков

Списки кроме стандартных данных типа строк, чисел, также могут содержать другие списки. Подобные списки можно ассоциировать с таблицами, где вложенные списки выполняют роль строк. Например:

```
users = [
    ["Tom", 29],
    ["Alice", 33],
    ["Bob", 27]
]

print(users[0])          # ["Tom", 29]
```

```
print(users[0][0])      # Tom
print(users[0][1])      # 29
```

Чтобы обратиться к элементу вложенного списка, необходимо использовать пару индексов: users[0][1] - обращение ко второму элементу первого вложенного списка.

Добавление, удаление и изменение общего списка, а также вложенных списков аналогично тому, как это делается с обычными (одномерными) списками:

```
users = [
    ["Tom", 29],
    ["Alice", 33],
    ["Bob", 27]
]

# создание вложенного списка
user = list()
user.append("Bill")
user.append(41)
# добавление вложенного списка
users.append(user)

print(users[-1]) # ["Bill", 41]

# добавление во вложенный список
users[-1].append("+79876543210")

print(users[-1]) # ["Bill", 41, "+79876543210"]

# удаление последнего элемента из вложенного списка
users[-1].pop()
print(users[-1]) # ["Bill", 41]

# удаление всего последнего вложенного списка
users.pop(-1)

# изменение первого элемента
users[0] = ["Sam", 18]
print(users)# [ ["Sam", 18], ["Alice", 33], ["Bob",
27]]
```

Кортеж

Кортеж(tuple) - представляет последовательность элементов, которая во многом похожа на список за тем исключением, что кортеж является неизменяемым (immutable) типом. Поэтому мы не можем добавлять или удалять элементы в кортеже, изменять его.

Для создания кортежа используются круглые скобки, в которые помещаются его значения, разделенные запятыми:

```
user = ("Tom", 23)
print(user)
```

Также для определения кортежа мы можем просто перечислить значения через запятую без применения скобок:

```
user = "Tom", 23
print(user)
```

Если вдруг кортеж состоит из одного элемента, то после единственного элемента кортежа необходимо поставить запятую:

```
user = ("Tom",)
```

Для создания кортежа из списка можно передать список в функцию tuple(), которая возвратит кортеж:

```
users_list = ["Tom", "Bob", "Kate"]
users_tuple = tuple(users_list)
print(users_tuple)      # ("Tom", "Bob", "Kate")
```

Обращение к элементам в кортеже происходит также, как и в списке по индексу. Индексация начинается также с нуля при получении элементов с начала списка и с -1 при получении элементов с конца списка:

```
users = ("Tom", "Bob", "Sam", "Kate")
print(users[0])     # Tom
print(users[2])     # Sam
```

```
print(users[-1])     # Kate

# получим часть кортежа со 2 элемента по 4
print(users[1:4])    # ("Bob", "Sam", "Kate")
```

Но так как кортеж - неизменяемый тип (immutable), то мы не сможем изменить его элементы. То есть следующая запись работать не будет:

```
users[1] = "Tim"
```

При необходимости мы можем разложить кортеж на отдельные переменные:

```
user = ("Tom", 22, False)
name, age, isMarried = user
print(name)              # Tom
print(age)               # 22
print(isMarried)         # False
```

Особенно удобно использовать кортежи, когда необходимо возвратить из функции сразу несколько значений. Когда функция возвращает несколько значений, фактически она возвращает в кортеж:

```
def get_user():
    name = "Tom"
    age = 22
    is_married = False
    return name, age, is_married

user = get_user()
print(user[0])               # Tom
print(user[1])               # 22
print(user[2])               # False
```

С помощью встроенной функции len() можно получить длину кортежа:

```
user = ("Tom", 22, False)
print(len(user))         # 3
```

Перебор кортежей

Для перебора кортежа можно использовать стандартные циклы for и while. С помощью цикла for:

```
user = ("Tom", 22, False)
for item in user:
    print(item)
С помощью цикла while:

user = ("Tom", 22, False)

i = 0
while i < len(user):
    print(user[i])
    i += 1
```

Как для списка с помощью выражения элемент in кортеж можно проверить наличие элемента в кортеже:

```
user = ("Tom", 22, False)
name = "Tom"
if name in user:
    print("Пользователя зовут Tom")
else:
    print("Пользователь имеет другое имя")
```

Словари

Наряду со списками и кортежами Python имеет еще одну встроенную структуру данных, которая называется словарь (dictionary). В ряде языков программирования есть похожие структуры (словарь в C#, ассоциативный массив в PHP).

Как и список, словарь хранит коллекцию элементов. Каждый элемент в словаре имеет уникальный ключ, с которым ассоциировано некоторое значение.

Определение словаря имеет следующий синтаксис:

```
dictionary = { ключ1:значение1, ключ2:значение2, ....}
```

Определим пару словарей:

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}
elements = {"Au": "Золото", "Fe": "Железо", "H": "Водород", "O": "Кислород"}
```

В словаре users в качестве ключей используются числа, а в качестве значений - строки. В словаре element в качестве ключей используются строки.

Но необязательно ключи и строки должны быть однотипными. Они могу представлять разные типы:

```
objects = {1: "Tom", "2": True, 3: 100.6}
```

Мы можем также вообще определить пустой словарь без элементов:

```
objects = {}
```

или так:

```
objects = dict()
```

Преобразование из списка в словарь

Несмотря на то, что словарь и список - непохожие по структуре типы, но тем не менее существует возможности для отдельных видов списков преобразования их в словарь с помощью встроенной функции dict(). Для этого список должен хранить набор вложенных списков. Каждый вложенный список должен состоять из двух элементов - при конвертации в словарь первый элемент станет ключом, а второй - значением:

```
users_list = [
    ["+111123455", "Tom"],
    ["+384767557", "Bob"],
    ["+958758767", "Alice"]
]
users_dict = dict(users_list)
```

```
print(users_dict)        #    {"+111123455":    "Tom",
"+384767557": "Bob", "+958758767": "Alice"}
```

Подобным образом можно преобразовать в словарь двухмерные кортежи, которые в свою очередь содержать кортежи из двух элементов:

```
users_tuple = (
    ("+111123455", "Tom"),
    ("+384767557", "Bob"),
    ("+958758767", "Alice")
)
users_dict = dict(users_tuple)
print(users_dict)
```

Получение и изменение элементов

Для доступа к элементам словаря необходимо использовать ключ:

```
dictionary[ключ]
```

Например, получим и изменим элементы в словаре:

```
users = {
    "+11111111": "Tom",
    "+33333333": "Bob",
    "+55555555": "Alice"
}

# получаем элемент с ключом "+11111111"
print(users["+11111111"])      # Tom

# установка значения элемента с ключом "+33333333"
users["+33333333"] = "Bob Smith"
print(users["+33333333"])      # Bob Smith
```

Если при установки значения элемента с таким ключом в словаре не окажется, то произойдет его добавление:

```
users["+4444444"] = "Sam"
```

Но если мы попробуем получить значение с ключом, которого нет в словаре, то Python сгенерирует ошибку KeyError:

```
user = users["+4444444"]     # KeyError
```

И чтобы предупредить эту ситуацию перед обращением к элементу мы можем проверять наличие ключа в словаре с помощью выражения ключ in словарь. Если ключ имеется в словаре, то данное выражение возвращает True:

```
key = "+4444444"
if key in users:
    user = users[key]
    print(user)
else:
    print("Элемент не найден")
```

Также для получения элементов можно использовать метод get, который имеет две формы:

1. get(key): возвращает из словаря элемент с ключом key. Если элемента с таким ключом нет, то возвращает значение None;
2. get(key, default): возвращает из словаря элемент с ключом key. Если элемента с таким ключом нет, то возвращает значение по умолчанию default.

```
key = "+55555555"
user = users.get(key)
user = users.get(key, "Unknown user")
```

Удаление

Для удаления элемента по ключу применяется оператор del:

```
users = {
    "+11111111": "Tom",
    "+33333333": "Bob",
    "+55555555": "Alice"
}

del users["+55555555"]
```

```
print(users)
```

Но стоит учитывать, что если подобного ключа не окажется в словаре, то будет выброшено исключение KeyError. Поэтому опять же перед удалением желательно проверять наличие элемента с данным ключом.

```
key = "+55555555"
if key in users:
    user = users[key]
    del users[key]
    print(user, "удален")
else:
    print("Элемент не найден")
```

Другой способ удаления представляет метод pop(). Он имеет две формы:

1. pop(key): удаляет элемент по ключу key и возвращает удаленный элемент. Если элемент с данным ключом отсутствует, то генерируется исключение KeyError
2. pop(key, default): удаляет элемент по ключу key и возвращает удаленный элемент. Если элемент с данным ключом отсутствует, то возвращается значение default

```
users = {
    "+11111111": "Tom",
    "+33333333": "Bob",
    "+55555555": "Alice"
}
key = "+55555555"
user = users.pop(key)
print(user)
user = users.pop("+4444444", "Unknown user")
print(user)
```

Если необходимо удалить все элементы, то в этом случае можно воспользоваться методом clear():

```
users.clear()
```

Копирование и объединение словарей

Метод copy() копирует содержимое словаря, возвращая новый словарь:

```
  users = {"+1111111": "Tom","+3333333":
"Bob","+5555555": "Alice"}
  users2 = users.copy()
  Метод update() объединяет два словаря:

  users = {"+1111111": "Tom","+3333333":
"Bob","+5555555": "Alice"}

  users2 = {"+2222222": "Sam","+6666666": "Kate"}
  users.update(users2)

  print(users)      # {"+1111111": "Tom", "+3333333":
"Bob",   "+5555555":   "Alice",   "+2222222":   "Sam",
"+6666666": "Kate"}
  print(users2)     # {"+2222222": "Sam", "+6666666":
"Kate"}
```

При этом словарь users2 остается без изменений. Изменяется словарь users, в который добавляются элементы другого словаря. Но если необходимо, чтобы оба исходных словаря были без изменений, а результатом объединения был какой-то третий словарь, то можно предварительно скопировать один словарь в другой:

```
  users3 = users.copy()
  users3.update(users2)
  Перебор словаря
  Для перебора словаря можно воспользоваться циклом
for:

  users = {
      "+11111111": "Tom",
      "+33333333": "Bob",
      "+55555555": "Alice"
  }
  for key in users:
```

```
        print(key, " - ", users[key])
```

При переборе элементов мы получаем ключ текущего элемента и по нему можем получить сам элемент.

Другой способ перебора элементов представляет использование метода items():

```
for key, value in users.items():
    print(key, " - ", value)
```

Метод items() возвращает набор кортежей. Каждый кортеж содержит ключ и значение элемента, которые при переборе мы тут же можем получить в переменные key и value.

Также существуют отдельно возможности перебора ключей и перебора значений. Для перебора ключей мы можем вызвать у словаря метод keys():

```
for key in users.keys():
    print(key)
```

Правда, этот способ перебора не имеет смысла, так как и без вызова метода keys() мы можем перебрать ключи, как было показано выше.

Для перебора только значений мы можем вызвать у словаря метод values():

```
for value in users.values():
    print(value)
```

Комплексные словари

Кроме простейших объектов типа чисел и строк словари также могут хранить и более сложные объекты - те же списки, кортежи или другие словари:

```
users = {
    "Tom": {
        "phone": "+971478745",
```

```
        "email": "tom12@gmail.com"
    },
    "Bob": {
        "phone": "+876390444",
        "email": "bob@gmail.com",
        "skype": "bob123"
    }
}
```

В данном случае значение каждого элемента словаря в свою очередь представляет отдельный словарь.Для обращения к элементам вложенного словаря соответственно необходимо использовать два ключа:

```
old_email = users["Tom"]["email"]
users["Tom"]["email"] = supertom@gmail.com
```

Но если мы попробуем получить значение по ключу, который отсутствует в словаре, Python сгенерирует исключение KeyError:

```
tom_skype = users["Tom"]["skype"]    # KeyError
```

Чтобы избежать ошибки, можно проверять наличие ключа в словаре:

```
key = "skype"
if key in users["Tom"]:
    print(users["Tom"]["skype"])
else:
    print("skype is not found")
```

Во всем остальном работа с комплексными и вложенными словарями аналогична работе с обычными словарями.

Множества.

Множество (set) представляют еще один вид набора элементов. Для определения множества используются фигурные скобки, в которых перечисляются элементы:

```
users = {"Tom","Bob","Alice", "Tom"}
```

```
print(users)     # {"Tom","Bob","Alice"}
```

Обратите внимание, что несмотря на то, что функция print вывела один раз элемент "Tom", хотя в определении множества этот элемент содержится два раза. Все потому что множество содержит только уникальные значения.

Также для определения множества может применяться функция set(), в которую передается список или кортеж элементов

```
users3 = set(["Mike", "Bill", "Ted"])
```

Функцию set удобно применять для создания пустого множества:

```
users = set()
```

Для получения длины множества применяется встроенная функция len():

```
users = {"Tom","Bob","Alice"}
print(len(users)}    # 3
```

Добавление элементов

Для добавления одиночного элемента вызывается метод add():

```
users = set()
users.add("Sam")
print(users)
```

Удаление элементов

Для удаления одного элемента вызывается метод remove(), в который передается удаляемый элемент. Но следует учитывать, что если такого элемента не окажется в множестве, то будет сгенерирована ошибка. Поэтому перед удалением следует проверять на наличие элемента с помощью оператора in:

```
users = {"Tom", "Bob", "Alice"}

user = "Tom"
```

```
if user in users:
    users.remove(user)
print(users)    # {"Bob", "Alice"}
```

Также для удаления можно использовать метод discard(), который не будет генерировать исключения при отсутствии элемента:

```
user = "Tim"
users.discard(user)
```

Для удаления всех элементов вызывается метод clear():

```
users.clear()
```

Перебор множества

Для перебора элементов можно использовать цикл for:

```
users = {"Tom","Bob","Alice"}

for user in users:
    print(user)
```

При переборе каждый элемент помещается в переменную user.

Операции с множествами

С помощью метода copy() можно скопировать содержимое одного множества в другую переменную:

```
users = {"Tom","Bob","Alice"}
users3 = users.copy()
```

Метод union() объединяет два множества и возвращает новое множество:

```
users = {"Tom","Bob","Alice"}
users2 = {"Sam","Kate", "Bob"}

users3 = users.union(users2)
print(users3)     # {"Bob", "Alice", "Sam", "Kate", "Tom"}
```

Пересечение множеств позволяет получить только те элементы, которые есть одновременно в обоих множествах. Метод intersection() производит операцию пересечения множеств и возвращает новое множество:

```
users = {"Tom","Bob","Alice"}
users2 = {"Sam","Kate", "Bob"}

users3 = users.intersection(users2)
print(users3)    # {"Bob"}
```

Вместо метода intersection мы могли бы использовать операцию логического умножения:

```
users = {"Tom","Bob","Alice"}
users2 = {"Sam","Kate", "Bob"}

print(users & users2)    # {"Bob"}
```

В этом случае мы получили бы тот же результат.

Еще одна операция - разность множеств возвращает те элементы, которые есть в первом множестве, но отсутствуют во втором. Для получения разности множеств можно использовать метод difference или операцию вычитания:

```
users = {"Tom","Bob","Alice"}
users2 = {"Sam","Kate", "Bob"}

users3 = users.difference(users2)
print(users3)            # {"Tom", "Alice"}
print(users - users2)    # {"Tom", "Alice"}
```

Отношения между множествами

Метод issubset позволяет выяснить, является ли текущее множество подмножеством (то есть частью) другого множества:

```
users = {"Tom", "Bob", "Alice"}
superusers = {"Sam", "Tom", "Bob", "Alice", "Greg"}
```

```
print(users.issubset(superusers))    # True
print(superusers.issubset(users))    # False
```

Метод issuperset, наоборот, возвращает True, если текущее множество является надмножеством (то есть содержит) для другого множества:

```
users = {"Tom", "Bob", "Alice"}
superusers = {"Sam", "Tom", "Bob", "Alice", "Greg"}

print(users.issuperset(superusers))    # False
print(superusers.issuperset(users))    # True
frozen set
```

Тип frozen set является видом множеств, которое не может быть изменено. Для его создания используется функция frozenset:

```
users = frozenset({"Tom", "Bob", "Alice"}
```

В функцию frozenset передается набор элементов - список, кортеж, другое множество.В такое множество мы не можем добавить новые элементы, как и удалить из него уже имеющиеся. Собственно поэтому frozen set поддерживает ограниченный набор операций:

1. len(s): возвращает длину множества
2. x in s: возвращает True, если элемент x присутствует в множестве s
3. x not in s: возвращает True, если элемент x отсутствует в множестве s
4. s.issubset(t): возвращает True, если t содержит множество s
5. s.issuperset(t): возвращает True, если t содержится в множестве s
6. s.union(t): возвращает объединение множеств s и t
7. s.intersection(t): возвращает пересечение множеств s и t
8. s.difference(t): возвращает разность множеств s и t
9. s.copy(): возвращает копию множества s

Работа с файлами. Открытие и закрытие файлов

Python поддерживает множество различных типов файлов, но условно их можно разделить на два виде: текстовые и бинарные. Текстовые файлы - это к примеру файлы с расширением cvs, txt, html, в общем любые файлы, которые сохраняют информацию в текстовом виде. Бинарные файлы - это изображения, аудио и видеофайлы и т.д. В зависимости от типа файла работа с ним может немного отличаться.

При работе с файлами необходимо соблюдать некоторую последовательность операций:

1. Открытие файла с помощью метода open()
2. Чтение файла с помощью метода read() или запись в файл посредством метода write()
3. Закрытие файла методом close()
4. Открытие и закрытие файла

Чтобы начать работу с файлом, его надо открыть с помощью функции open(), которая имеет следующее формальное определение:

```
open(file, mode)
```

Первый параметр функции представляет путь к файлу. Путь файла может быть абсолютным, то есть начинаться с буквы диска, например, C://somedir/somefile.txt. Либо можно быть относительным, например, somedir/somefile.txt - в этом случае поиск файла будет идти относительно расположения запущенного скрипта Python.

Второй передаваемый аргумент - mode устанавливает режим открытия файла в зависимости от того, что мы собираемся с ним делать. Существует 4 общих режима:

1. r (Read). Файл открывается для чтения. Если файл не найден, то генерируется исключение FileNotFoundError
2. w (Write). Файл открывается для записи. Если файл отсутствует, то он создается. Если подобный файл уже есть, то он создается заново, и соответственно старые данные в нем стираются.
3. a (Append). Файл открывается для дозаписи. Если файл отсутствует, то он создается. Если подобный файл уже есть, то данные записываются в его конец.

4. b (Binary). Используется для работы с бинарными файлами. Применяется вместе с другими режимами - w или r.

После завершения работы с файлом его обязательно нужно закрыть методом close(). Данный метод освободит все связанные с файлом используемые ресурсы.

Например, откроем для записи текстовый файл "hello.txt":

```
myfile = open("hello.txt", "w")

myfile.close()
```

При открытии файла или в процессе работы с ним мы можем столкнуться с различными исключениями, например, к нему нет доступа и т.д. В этом случае программа выпадет в ошибку, а ее выполнение не дойдет до вызова метода close, и соответственно файл не будет закрыт.

В этом случае мы можем обрабатывать исключения:

```
try:
    somefile = open("hello.txt", "w")
    try:
        somefile.write("hello world")
    except Exception as e:
        print(e)
    finally:
        somefile.close()
except Exception as ex:
    print(ex)
```

В данном случае вся работа с файлом идет во вложенном блоке try. И если вдруг возникнет какое-либо исключение, то в любом случае в блоке finally файл будет закрыт.

Однако есть и более удобная конструкция - конструкция with:

```
with open(file, mode) as file_obj:
    инструкции
```

Эта конструкция определяет для открытого файла переменную file_obj и выполняет набор инструкций. После их выполнения файл автоматически закрывается. Даже если при выполнении инструкций в блоке with возникнут какие-либо исключения, то файл все равно закрывается.

Так, перепишем предыдущий пример:

```
with open("hello.txt", "w") as somefile:
    somefile.write("hello world")
```

Запись в текстовый файл

Чтобы открыть текстовый файл на запись, необходимо применить режим w (перезапись) или a (дозапись). Затем для записи применяется метод write(str), в который передается записываемая строка. Стоит отметить, что записывается именно строка, поэтому, если нужно записать числа, данные других типов, то их предварительно нужно конвертировать в строку.

Запишем некоторую информацию в файл "hello.txt":

```
with open("hello.txt", "w") as file:
    file.write("hello world")
```

Если мы откроем папку, в которой находится текущий скрипт Python, то увидем там файл hello.txt. Этот файл можно открыть в любом текстовом редакторе и при желании изменить.

Теперь дозапишем в этот файл еще одну строку:

```
with open("hello.txt", "a") as file:
    file.write("\ngood bye, world")
```

До запись выглядит как добавление строку к последнему символу в файле, поэтому, если необходимо сделать запись с новой строки, то можно использовать эскейп-последовательность "\n". В итоге файл hello.txt будет иметь следующее содержимое:

```
hello world
good bye, world
```

Еще один способ записи в файл представляет стандартный метод print(), который применяется для вывода данных на консоль:

```
with open("hello.txt", "a") as hello_file:
    print("Hello, world", file=hello_file)
```

Для вывода данных в файл в метод print в качестве второго параметра передается название файла через параметр file. А первый параметр представляет записываемую в файл строку.

Чтение файла

Для чтения файла он открывается с режимом r (Read), и затем мы можем считать его содержимое различными методами:

1. readline(): считывает одну строку из файла
2. read(): считывает все содержимое файла в одну строку
3. readlines(): считывает все строки файла в список

Например, считаем выше записанный файл построчно:

```
with open("hello.txt", "r") as file:
    for line in file:
        print(line, end="")
```

Несмотря на то, что мы явно не применяем метод readline() для чтения каждой строки, но в при переборе файла этот метод автоматически вызывается для получения каждой новой строки. Поэтому в цикле вручную нет смысла вызывать метод readline. И поскольку строки разделяются символом перевода строки "\n", то чтобы исключить излишнего переноса на другую строку в функцию print передается значение end="".

Теперь явным образом вызовем метод readline() для чтения отдельных строк:

```
with open("hello.txt", "r") as file:
    str1 = file.readline()
    print(str1, end="")
    str2 = file.readline()
    print(str2)
```

Метод readline можно использовать для построчного считывания файла в цикле while:

```
with open("hello.txt", "r") as file:
    line = file.readline()
    while line:
        print(line, end="")
        line = file.readline()
```

Если файл небольшой, то его можно разом считать с помощью метода read():

```
with open("hello.txt", "r") as file:
    content = file.read()
    print(content)
```

И также применим метод readlines() для считывания всего файла в список строк:

```
with open("hello.txt", "r") as file:
    contents = file.readlines()
    str1 = contents[0]
    str2 = contents[1]
    print(str1, end="")
    print(str2)
```

Работа со строками

Строка представляет последовательность символов в кодировке Unicode. И мы можем обратиться к отдельным символам строки по индексу в квадратных скобках:

```
string = "hello world"
c0 = string[0]  # h
print(c0)
c6 = string[6]  # w
print(c6)

c11 = string[11]   # ошибка IndexError: string index
out of range
print(c11)
```

Индексация начинается с нуля, поэтому первый символ строки будет иметь индекс 0. А если мы попытаемся обратиться к индексу, которого нет в строке, то мы получим исключение IndexError. Например, в случае выше длина строки 11 символов, поэтому ее символы будут иметь индексы от 0 до 10.

Чтобы получить доступ к символам, начиная с конца строки, можно использовать отрицательные индексы. Так, индекс -1 будет представлять последний символ, а -2 - предпоследний символ и так далее:

```
string = "hello world"
c1 = string[-1]  # d
print(c1)
c5 = string[-5]  # w
print(c5)
```

При работе с символами следует учитывать, что строка - это неизменяемый (immutable) тип, поэтому если мы попробуем изменить какой-то отдельный символ строки, то мы получим ошибку, как в следующем случае:

```
string = "hello world"
string[1] = "R"
```

Мы можем только полностью переустановить значение строки, присвоив ей другое значение.

Получение подстроки

При необходимости мы можем получить из строки не только отдельные символы, но и подстроку. Для этого используется следующий синтаксис:

1. string[:end]: извлекается последовательность символов начиная с 0-го индекса по индекс end
2. string[start:end]: извлекается последовательность символов начиная с индекса start по индекс end

3. string[start:end:step]: извлекается последовательность символов начиная с индекса start по индекс end через шаг step

Используем все варианты получения подстроки:

```
string = "hello world"

# с 0 до 5 символа
sub_string1 = string[:5]
print(sub_string1)      # hello

# со 2 до 5 символа
sub_string2 = string[2:5]
print(sub_string2)      # llo

# со 2 по 9 символ через один символ
sub_string3 = string[2:9:2]
print(sub_string3)      # lowr
```

Функции ord и len

Поскольку строка содержит символы Unicode, то с помощью функции ord() мы можем получить числовое значение для символа в кодировке Unicode:

```
print(ord("A")) # 65
```

Для получения длины строки можно использовать функцию len():

```
string = "hello world"
length = len(string)
print(length)   # 11
```

Поиск в строке

С помощью выражения term in string можно найти подстроку term в строке string. Если подстрока найдена, то выражение вернет значение True, иначе возвращается значение False:

```
string = "hello world"
exist = "hello" in string
print(exist)    # True

exist = "sword" in string
```

```
print(exist)     # False
```

Перебор строки

С помощью цикла for можно перебрать все символы строки:

```
string = "hello world"
for char in string:
    print(char)
```

Основные методы строк

Рассмотрим основные методы строк, которые мы можем применить в приложениях:

1. isalpha(str): возвращает True, если строка состоит только из алфавитных символов
2. islower(str): возвращает True, если строка состоит только из символов в нижнем регистре
3. isupper(str): возвращает True, если все символы строки в верхнем регистре
4. isdigit(str): возвращает True, если все символы строки - цифры
5. isnumeric(str): возвращает True, если строка представляет собой число
6. startwith(str): возвращает True, если строка начинается с подстроки str
7. endwith(str): возвращает True, если строка заканчивается на подстроку str
8. lower(): переводит строку в нижний регистр
9. upper(): переводит строку в вехний регистр
10. title(): начальные символы всех слов в строке переводятся в верхний регистр
11. capitalize(): переводит в верхний регистр первую букву только самого первого слова строки
12. lstrip(): удаляет начальные пробелы из строки
13. rstrip(): удаляет конечные пробелы из строки
14. strip(): удаляет начальные и конечные пробелы из строки

15. ljust(width): если длина строки меньше параметра width, то справа от строки добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по левому краю
16. rjust(width): если длина строки меньше параметра width, то слева от строки добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по правому краю
17. center(width): если длина строки меньше параметра width, то слева и справа от строки равномерно добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по центру
18. find(str[, start [, end]): возвращает индекс подстроки в строке. Если подстрока не найдена, возвращается число -1
19. replace(old, new[, num]): заменяет в строке одну подстроку на другую
20. split([delimeter[, num]]): разбивает строку на подстроки в зависимости от разделителя
21. join(strs): объединяет строки в одну строку, вставляя между ними определенный разделитель

Например, если мы ожидаем ввод с клавиатуры числа, то перед преобразованием введенной строки в число можно проверить, с помощью метода isnumeric() введено ли в действительности число, и если так, то выполнить операцию преобразования:

```
cstring = input("Введите число: ")
if string.isnumeric():
    number = int(string)
    print(number)
```

Проверка, начинается или оканчивается строка на определенную подстроку:

```
file_name = "hello.py"
starts_with_hello = file_name.startswith("hello")    # True
ends_with_exe = file_name.endswith("exe")            # False
```

Удаление пробелов в начале и в конце строки:

```
string = "   hello  world!  "
string = string.strip()
print(string)           # hello  world!
```

Дополнение строки пробелами и выравнивание:

```
print("iPhone 7:", "52000".rjust(10))
print("Huawei P10:", "36000".rjust(10))
```

Поиск в строке

Для поиска подстроки в строке в Python применяется метод find(), который возвращает индекс первого вхождения подстроки в строку и имеет три формы:

1. find(str): поиск подстроки str ведется с начала строки до ее конца
2. find(str, start): параметр start задает начальный индекс, с которого будет производиться поиск
3. find(str, start, end): параметр end задает конечный индекс, до которого будет идти поиск

Если подстрока не найдена, метод возвращает -1:

```
welcome = "Hello world! Goodbye world!"
index = welcome.find("wor")
print(index)        # 6

# поиск с 10-го индекса
index = welcome.find("wor",10)
print(index)        # 21

# поиск с 10 по 15 индекс
index = welcome.find("wor",10,15)
print(index)        # -1
```

Замена в строке

Для замены в строке одной подстроки на другую применяется метод replace():

1. replace(old, new): заменяет подстроку old на new
2. replace(old, new, num): параметр num указывает, сколько вхождений подстроки old надо заменить на new

```
phone = "+1-234-567-89-10"

# замена дефисов на пробел
edited_phone = phone.replace("-", " ")
print(edited_phone)      # +1 234 567 89 10

# удаление дефисов
edited_phone = phone.replace("-", "")
print(edited_phone)      # +12345678910

# замена только первого дефиса
edited_phone = phone.replace("-", "", 1)
print(edited_phone)      # +1234-567-89-10
```

Разделение на подстроки

Метод split() разбивает строку на список подстрок в зависимости от разделителя. В качестве разделителя может выступать любой символ или последовательность символов. Данный метод имеет следующие формы:

1. split(): в качестве разделителя используется пробел
2. split(delimeter): в качестве разделителя используется delimeter
3. split(delimeter, num): параметр num указывает, сколько вхождений delimeter используется для разделения. Оставшаяся часть строки добавляется в список без разделения на подстроки

```
text = "Это был огромный, в два обхвата дуб, с обломанными ветвями и с обломанной корой"
# разделение по пробелам
splitted_text = text.split()
print(splitted_text)
print(splitted_text[6])      # дуб,

# разбиение по запятым
splitted_text = text.split(",")
print(splitted_text)
print(splitted_text[1])      # в два обхвата дуб

# разбиение по первым пяти пробелам
```

```
splitted_text = text.split(" ", 5)
print(splitted_text)
print(splitted_text[5])      # обхвата дуб, с обломанными
ветвями и с обломанной корой
```

Соединение строк

При рассмотрении простейших операций со строками было показано, как объединять строки с помощью операции сложения. Другую возможность для соединения строк представляет метод join(): он объединяет список строк. Причем текущая строка, у которой вызывается данный метод, используется в качестве разделителя:

```
words = ["Let", "me", "speak", "from", "my", "heart",
"in", "English"]

# разделитель - пробел
sentence = " ".join(words)
print(sentence)  # Let me speak from my heart in English

# разделитель - вертикальная черта
sentence = " | ".join(words)
print(sentence)  # Let | me | speak | from | my | heart
| in | English
```

Вместо списка в метод join можно передать простую строку, тогда разделитель будет вставляться между символами этой строки:

```
word = "hello"
joined_word = "|".join(word)
print(joined_word)      # h|e|l|l|o
```

ЗАДАЧИ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Разберем на конкретных примерах процесс реализации задач для выполнения лабораторной работы.

Первое задание. Создать два множества: книг на семестр и книг, имеющихся в библиотеке. Создать список необходимых книг, которые можно получить в данной библиотеке, и список книг, которых нет в наличии. Вывести оба списка на экран.

Сначала создадим два множества-множество read(множество книг на семестр) и множество library(множество книг, имеющихся в библиотеке).Заполним их в процессе написание множеств.Итого у нас получится:

```
must_read = set([
    'Английский Вишневская',
    'Дискретная математика Новиков',
    'Математика Иванов',
    'Язык С Карнеги',
    'Ассемблер Яснов'
])

library = set([
    'Английский Вишневская',
    'Математика Иванов',
    'Ассемблер Яснов',
    'Искусствоведение Клементовская',
    'Русский язык Анашина'
])
```

Затем объедим два действия: вывод и создание списков книг, которых нет в наличии и книг, которых можно получить в данной библиотеке:

```
  print('Можно получить: ', ', '.join(list(must_read
& library)) )
  print('Нет в наличии: ', ', '.join(list(must_read -
library)) )
```

Итоговый листинг будет выглядеть следующим образом:

```
must_read = set([
    'Английский Вишневская',
    'Дискретная математика Новиков',
    'Математика Иванов',
    'Язык С Карнеги',
    'Ассемблер Яснов'
])

library = set([
    'Английский Вишневская',
    'Математика Иванов',
    'Ассемблер Яснов',
    'Искусствоведение Клементовская',
    'Русский язык Анашина'
])

print('Можно получить: ', ', '.join(list(must_read
& library)) )
print('Нет в наличии: ', ', '.join(list(must_read -
library)) )
```

Тестирование:

```
Можно получить:  Ассемблер Яснов, Английский
Вишневская, Математика Иванов
Нет в наличии:  Дискретная математика Новиков, Язык С
Карнеги
```

Задание второе. Считать файл, содержащий информации о книгах: название и год издания. В файл k вывести название книг, изданных не позже 1990 года.

Сначала откроем оба файла, затем в цикле while будем проходить по каждому компоненту файла и выводить название только тех книг, год издания которых не превышает 1990. Пример реализации:

```
file_input = open('task2_in.txt', 'r')
file_output = open('task2_out.txt', 'w')

while True:
```

```
    title = file_input.readline()
    if title == '':
        break
    year = int( file_input.readline() )

    if year <= 1990:
        file_output.write(title)
```

Тестирование:

Данные, находящиеся в файле:

```
451° по Фаренгейту
1953
Гарри Поттер и Дары Смерти
2007
Мечтают ли андроиды об электроовцах?
1968
1984
1948
Гарри Поттер и Орден Феникса
2003
Три товарища
1936
2001: Космическая одиссея
1968
```

Данные, которые выводятся в новый файл согласно условию задачи:

```
451° по Фаренгейту
Мечтают ли андроиды об электроовцах?
1984
Три товарища
2001: Космическая одиссея
```

Задание третье. Создать словарь, состоящий из имен и дат рождения (ключ — имя), вводимые пользователем с клавиатуры. Создать файл, каждая строчка которого будет содержать имя и дату рождения.

Приведем листинг одного из возможных вариантов решения данной задачи:

```
data = {}

n = int(input("Количество имен: "))

for i in range(n):
```

```
    name = input("Имя: ")
    date = input("Дата рождения: ")

    data[name] = date

file_output = open('task3_out.txt', 'w')

for name in data:
    file_output.write(name + ' ' + data[name] + '\n')
```

Тестирование:

```
Количество имен: 5
Имя: Женя
Дата рождения: 23.10.99
Имя: Кирилл
Дата рождения: 15.01.98
Имя: Антон
Дата рождения: 12.04.66
Имя: Анжелика
Дата рождения: 08.09.14
Имя: Гарри
Дата рождения: 31.08.99
Женя 23.10.99
Кирилл 15.01.98
Антон 12.04.66
Анжелика 08.09.14
Гарри 31.08.99
```

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Вариант 1

Задание 1. Дана строка, содержащая русскоязычный текст. Найти количество слов, начинающихся с буквы "е".

Задание 2. Из массива X длиной n, среди элементов которого есть положительные, отрицательные и равные нулю, сформировать новый массив Y, взяв в него только те элементы из X, которые больше по модулю заданного числа M. Вывести на экран число M, данный и полученные массивы.

Задание 3. В массиве целых чисел все отрицательные элементы заменить на положительные. Вывести исходный массив и полученный.

Задание 4. Создать функцию, возвращающую 3 параметра (имя, пол и профессия, значения считываются с клавиатуры). С помощью таких функций создать несколько кортежей и построить на их основе словарь, где ключами являются имена, а значениями их профессии. Вывести словарь на экран

Вариант 2

Задание 1. В строке заменить все двоеточия (:) знаком процента (%). Подсчитать количество замен.

Задание 2. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти максимальный элемент. Вывести массив на экран в обратном порядке.

Задание 3. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое всех элементов массива.

Задание 4. Создать словарь — телефонную книгу. Пользователь вводит с клавиатуры количество контактов, и по очереди их имена и номера. Из этого словаря получить список, содержащий только имена и вывести его.

Вариант 3

Задание 1. В строке удалить символ точку (.) и подсчитать количество удаленных символов.

Задание 2. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти минимальный элемент. Вывести индекс минимального элемента на экран.

Задание 3. Дан массив целых чисел. Переписать все положительные элементы во второй массив, а остальные - в третий.

Задание 4. Создать 2 множества, одно содержит людей, знакомых Полу, другое — людей, знакомых Мари. Оба заполняются вручную с клавиатуры. Получить список общих знакомых, список людей, знакомых только для Пола, и список людей, знакомых только для Мари, и список знакомых хотя бы для одного из них. Вывести все 4 списка.

Вариант 4

Задание 1. В строке заменить букву(а) буквой (о). Подсчитать количество замен. Подсчитать, сколько символов в строке.

Задание 2. В одномерном числовом массиве D длиной n вычислить сумму элементов с нечетными индексами. Вывести на экран массив D, полученную сумму.

Задание 3. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Задание 4. Создать 3 множества — предметы, изучаемые на 3 направлениях. Создать 4 списка: Список предметов общих для всех направлений и по одному списку — предметы, которые есть только в данном направлении. Вывести все списки на экран.

Вариант 5

Задание 1. В строке заменить все заглавные буквы строчными.

Задание 2. Дан массив целых чисел. Найти максимальный элемент массива и его порядковый номер.

Задание 3. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Задание 4. Автоматически (не с клавиатуры) создать несколько кортежей, содержащих имя, возраст и номер телефона. Разложить их на переменные и получить из них 2 списка: список имен и список номеров. Вывести их на экран.

Вариант 6

Задание 1. В строке удалить все буквы "а" и подсчитать количество удаленных символов.

Задание 2. Дан одномерный массив из 10 целых чисел. Вывести пары отрицательных чисел, стоящих рядом.

Задание 3. Дан целочисленный массив размера 10. Создать новый массив, удалив все одинаковые элементы, оставив их 1 раз.

Задание 4. Создать словарь товаров. Ключами являются названия товаров, а значениями — их цены. Пользователь вводит количество товаров, которые хочет купить, и их наименования. Посчитать и вывести на экран общую сумму покупки. Если такого товара нет, то вывести об этом сообщение и запросить другое название.

Вариант 7

Задание 1. Дана строка. Преобразовать ее, заменив звездочками все буквы "п", встречающиеся среди первых n/2 символов. Здесь n - длина строки.

Задание 2. Дан одномерный массив из 10 целых чисел. Найти максимальный элемент и сравнить с ним остальные элементы. Вывести количество меньших максимального и больших максимального элемента.

Задание 3. Одномерный массив из 10-и целых чисел заполнить с клавиатуры, определить сумму тех чисел, которые >5.

Задание 4. Создать словарь. Пользователь вводит с клавиатуры количество контактов, и по очереди их имена и даты рождения. После пользователь вводит дату рождения. Программа должна создать список

людей, родившихся в этот день и вывести его на экран. Если таких людей нет, то вывести такое сообщение.

Вариант 8

Задание 1. Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.

Задание 2. Дан массив целых чисел. Найти сумму элементов с четными номерами и произведение элементов с нечетными номерами. Вывести сумму и произведение.

Задание 3. Переставить в одномерном массиве минимальный элемент и максимальный.

Задание 4. Создать 2 множества: необходимых для учёбы вещей и вещей, находящихся в сумке у Мари. Оба заполняются вручную. Создать списки вещей, которых Мари не хватает, и ненужных вещей в сумке Мари. Вывести их на экран.

Вариант 9

Задание 1. Определить, сколько раз в тексте встречается заданное слово.

Задание 2. Найдите сумму и произведение элементов списка. Результаты вывести на экран.

Задание 3. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое всех элементов массива.

Задание 4. Создать словарь товаров. Ключами являются названия товаров, а значениями — их цены. Пользователь вводит количество товаров, которые хочет купить, и их наименования. Посчитать и вывести на экран общую сумму покупки. Если такого товара нет, то вывести об этом сообщение и запросить другое название.

Вариант 10

Задание 1. Дана строка-предложение на английском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы.

Задание 2. Дан одномерный массив, состоящий из N вещественных элементов. Ввести массив с клавиатуры. Найти и вывести минимальный по модулю элемент. Вывести массив на экран в обратном порядке.

Задание 3. Даны массивы A и B одинакового размера 10. Вывести исходные массивы. Поменять местами их содержимое и вывести в начале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Задание 4. Создать кортеж, в котором чередуются имена и возраста людей (имена не могут повторяться. Прим: Пол, 15, Мари, 18, Питер, 16). Из него получить и вывести словарь, ключами которого являются имена, а значениями — их возраст.

Вариант 11

Задание 1. Дана строка. Подсчитать самую длинную последовательность подряд идущих букв «н». Преобразовать ее, заменив точками все восклицательные знаки.

Задание 2. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран это значение, иначе сообщение об их отсутствии.

Задание 3. Дан одномерный массив из 15 элементов. Элементам массива меньше 10 присвоить нулевые значения, а элементам больше 20 присвоить 1. Вывести на экран монитора первоначальный и преобразованный массивы в строчку.

Задание 4. Создать словарь — телефонную книгу. Пользователь вводит с клавиатуры количество контактов, и по очереди их имена и номера. Из этого словаря получить список, содержащий только имена и вывести его.

Вариант 12

Задание 1. Дана строка. Вывести все слова, оканчивающиеся на букву "я".

Задание 2. Найти наибольший элемент списка, который делиться на 2 без остатка и вывести его на экран.

Задание 3. Дан одномерный массив целого типа. Получить другой массив, состоящий только из четных чисел исходного массива, меньше 10, или сообщить, что таких чисел нет. Полученный массив вывести в порядке возрастания элементов.

Задание 4. Создать функцию, возвращающую 3 параметра (имя, пол и профессия, значения считываются с клавиатуры). С помощью таких функций создать несколько кортежей и построить на их основе словарь, где ключами являются имена, а значениями их профессии. Вывести словарь на экран

Вариант 13

Задание 1. Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобки. Вывести на экран все символы, расположенные внутри этих скобок.

Задание 2. Найти наименьший нечетный элемент списка и вывести его на экран.

Задание 3. Даны массивы A и B одинакового размера 10. Поменять местами их содержимое и вывести вначале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Задание 4. Вручную создать два множества: игрушки, имеющиеся в данном магазине, и игрушки фирмы Haribo. Получить список игрушек этой фирмы, которых нет в магазине, и список игрушек данного магазина, произведенные другими фирмами. Вывести оба списка на экран.

Вариант 14

Задание 1. Дана строка. Вывести все слова, начинающиеся на букву "а" и слова оканчивающиеся на букву "я".

Задание 2. Дан одномерный массив целых чисел. Проверить, есть ли в нем одинаковые элементы. Вывести эти элементы и их индексы.

Задание 3. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Задание 4. Создать словарь, содержащий в качестве ключей названия приложений и в качестве значений - их вес. Пользователь указывает определённое значение, и программа создаёт и выводит на экран список приложений, вес которых меньше введённого значения.

Вариант 15

Задание 1. Дана строка текста. Подсчитать количество букв «т» в строке.

Задание 2. Найти максимальный элемент численного массива и поменять его местами с минимальным.

Задание 3. Программа заполняет одномерный массив из 10 целых чисел числами, считанными с клавиатуры. Определить среднее арифметическое всех чисел массива. Заменить элементы массива большие среднего арифметического на 1.

Задание 4. Вручную создать два множества: игрушки, имеющиеся в данном магазине, и игрушки фирмы Haribo. Получить список игрушек этой фирмы, которых нет в магазине, и список игрушек данного магазина, произведенные другими фирмами. Вывести оба списка на экран.

Вариант 16

Задание 1. Проверить, будет ли строка читаться одинаково справа налево и слева направо (т. е. является ли она палиндромом).

Задание 2. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран эти значения.

Задание 3. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Задание 4. Создать словарь, содержащий в качестве ключей названия приложений и в качестве значений - их вес. Пользователь указывает определённое значение, и программа создаёт и выводит на экран список приложений, вес которых меньше введённого значения.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка заданий.
4. Блок-схемы для каждого задания.
5. Листинг программы для каждого задания.
6. Результаты выполнения программ.
7. Выводы по работе в целом.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Расскажите о компонентах сложных структур данных Python.
2. Объясните, что представляет собой список.
3. Как создать список, в котором необходимо реализовать повторение одно и то же значение несколько раз?
4. Как происходит копирование части списка?
5. Сформулируйте определение кортежей.
6. Расскажите, какие основные методы работы с кортежами были изучены в данной лабораторной работе.
7. Что представляет собой множество в Python?
8. Перечислите методы работы с элементами в множестве, которые были изучены.
9. Опишите процесс работы с файлами(как происходит открытие и закрытие).
10. Назовите два основных типа файлов, которые поддерживает Python.
11. Охарактеризуйте работу со строкой в Python.
12. Расскажите основные функции обработки строк в Python.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Разработать программы согласно варианту.
4. Выполнить тестирование программ.
5. Продемонстрировать работу программ преподавателю.
6. Оформить отчет.
7. Защитить выполненную работу у преподавателя.

ЛАБОРАТОРНАЯ РАБОТА №3
ОБЪЕКТНО-ОРИИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА PYTHON

Целью выполнения лабораторной работы является формирование практических навыков процедурного программирования, разработки и отладки программ, овладение методами и средствами разработки и оформления технической документации.

Основными задачами выполнения лабораторной работы являются:

1. Изучить особенности создания классов;
2. Научиться создавать экземпляры классов;
3. Изучить типовые алгоритмы решения задач с использованием принципов объектно-ориентированного программирования.

Результатами работы являются:

1. Реализация разработанных алгоритмов на языке программирования Python;
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Объектно-ориентированное программирование (в дальнейшем ООП) — это предназначенная для снижения сложности и повышения надежности методика разработки программ, в которой основными понятиями являются классы и объекты.

В языке Python класс равносилен понятию тип данных. Он описывает группы однородных объектов: их характеристики и возможные действия. Например, тип int – это класс целых чисел. Числа 2, 25, 2019, -10 и т. д. – это конкретные объекты этого класса.

Python поддерживает создание классов и работу с ними в одной рабочей вкладке. Для объявление класса в Python используется ключевое слово class, после которого указывается идентификатор (имя класса) и ставится двоеточие. С новой строки начинается описание тела класса, в котором перечисляются атрибуты (переменные) и указываются методы. Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков. Рекомендуется использовать отступы, равные величине 4 пробелов или табуляции.

Переменные, определённые внутри класса (не метода), называются переменными экземпляра или полями (fields).

Метод — это именованный блок кода, который объявляется внутри класса и может быть использован многократно. Для его объявления необходимо использовать зарезервированное слово def, после которого указывается название метода и в скобках его параметры. Отметим, что первым параметром каждого метода на языке Python всегда является self (ключевое слово, которое относится к данному классу). Оно необходимо, чтобы разграничить поля и параметры.

Приведём пример создания содержащего атрибуты и метод класса:

```
class Robot:
    x = 5   # integer
    y = 5.0 # real
    def forvard(self, x, y):
        self.x += x
        self.y += y
```

Отметим, что классы принято называть с заглавной буквы, используя стиль UpperCamelCase, в то время как при именовании атрибутов и методов следует придерживаться стиля lowerCamelCasea. В классе могут быть несколько переменных и методов.

Конкретный представитель класса называется объектом. Он наследует свойства и методы класса. В языке программирования Python объекты принято называть также экземплярами. Это связано с тем, что в нем все классы сами являются объектами класса type. Точно также как все модули являются объектами класса module. Поэтому во избежание путаницы объекты, созданные на основе обычных классов, называют экземплярами.

```
#Создаём объект класса Robot
wally = Robot()
```

Конструктор

Кроме обычных методов классы могут определять специальные методы, которые называются конструкторами. Они вызываются при создании нового объекта данного класса и выполняют инициализацию объекта. В отличие от стандартных методов, конструктор никогда ничего не возвращает.

Если в классе не определено ни одного конструктора, то для этого класса автоматически создается конструктор без параметров. Имя такого метода обычно регламентируется синтаксисом конкретного языка программирования. Так в Java имя конструктора класса совпадает с именем самого класса. В Python же роль конструктора играет метод __init__().

Конструктор определяет действия, выполняемые при создании объекта класса, и является важной частью класса. Как правило, программисты стараются явно указать конструктор. Если явного конструктора нет, то Python автоматически создаст его для использования по умолчанию.

Определенный выше определенный класс Robot не имеет явно определённого конструктора. Поэтому для него автоматически создается конструктор по умолчанию, который мы можем использовать для создания класса объекта Robot.

Наличие пар знаков подчеркивания спереди и сзади в имени метода __init__ говорит о том, что он принадлежит к группе методов перегрузки операторов. Если подобные методы определены в классе, то объекты могут участвовать в таких операциях как сложение, вычитание, вызываться как функции и др.

Рассмотрим пример использования __init__() в явно определённых конструкторах:

```
class Person:
    def __init__(self, name, surname):
        self.name = name
        self.surname = surname
p1 = Person("Иванов", "Иван")
print(p1.name, p1.surname)
```

Результат: Иванов Иван

При этом методы перегрузки операторов не надо вызывать по имени. Вызовом для них является сам факт участия объекта в определенной операции. В случае конструктора класса – это операция создания объекта. Так как объект создается в момент вызова класса по имени, то в этот момент вызывается метод __init__(), если он определен в классе.

Деструктор

Помимо конструктора объектов в языках программирования есть обратный ему метод – деструктор. Он вызывается, когда объект не создается, а уничтожается.

В языке программирования Python объект уничтожается, когда исчезают все связанные с ним переменные или им присваивается другое значение, в результате чего связь со старым объектом теряется. Удалить переменную можно с помощью команды языка del. В классах Python функцию деструктора выполняет метод __del__(). После окончания работы с объектом мы можем использовать оператор **del** для удаления его из памяти:

```
person  = Person("Tom")
del person  # удаление из памяти
```

Основные принципы ООП

Основными принципами структурирования, которые придают объектам новые свойства, в объектно-ориентированных языках являются: абстрагирование, наследование, инкапсуляция и полиморфизм.

Инкапсуляция

Инкапсуляция — ограничение доступа к составляющим объект компонентам (методам и переменным). Инкапсуляция делает некоторые из компонент доступными только внутри класса.

Однако инкапсуляция в Python работает лишь на уровне соглашения между программистами о том, какие атрибуты являются общедоступными, а какие — внутренними. Таким образом, одиночное подчеркивание в начале имени атрибута говорит о том, что переменная или метод не предназначен для использования вне методов класса, однако атрибут доступен по этому имени.

Двойное подчеркивание в начале имени атрибута даёт большую защиту: атрибут становится недоступным по этому имени, то есть мы сможем обратиться к нему только из того же класса, где он определён, но не сможем обратиться к нему вне этого класса.

Приведём пример создания методов с ограниченным доступом:

```
class Car:
    # при вызове данного метода будет ошибка
```

```
    def _private(self):
        print("Это приватный метод!")
    # при вызове данного метода будет ошибка
    def __private(self):
        print("Это приватный метод!")
car = Car()
car._private()
car.__private()
```

Рекомендуется как можно больше ограничивать доступ к данным, чтобы защитить их от нежелательного доступа извне (как для получения значения, так и для его изменения). Использование различных модификаторов гарантирует, что данные не будут искажены или изменены не надлежащим образом.

Наследование - один из четырёх важнейших механизмов объектно-ориентированного программирования, позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса, доступ к которым не был ограничен двойным подчёркиванием, заимствуются новым классом.

Термин “родительский класс” равнозначен таким понятиям, как “базовый класс” и “суперкласс”. Класс-наследник, в свою очередь, можно называть подклассом, производным или дочерним классом.

Чтобы объявить один класс наследником от другого, необходимо при создании нового класса после имени указать в круглых скобках название суперкласса, от которого будут наследоваться поля и методы:

```
class Car (Vehicle):
    # методы и поля
```

Концепция наследования в программировании позволяет сократить время разработки, упростить процесс написания программы как сейчас, так и в будущем, когда заказчик захочет внести "небольшие правки" в проект. Правильно спроектированный класс это, кроме прочего, гибкая структура, которую вы свободно сможете изменять, дополнять в будущем.

Полиморфизм

Полиморфизм в объектно-ориентированном программировании реализует возможность обработки данных, принадлежащих к разным классам, с помощью функции или метода, которые были унаследованы от базового класса. Результат работы одноименных методов может существенно различаться, поэтому в данном контексте под полиморфизмом понимается множество форм одного и того же слова – имени метода.

Когда несколько классов имеют одни и те же имена методов, но различные реализации для этих же методов, классы являются полиморфными. Приведём пример:

```
class Vihicle ():
    def printType(self):
        print("Наземный вид транспорта")
    def printName(self):
        print("Транспортное средство")
class Car(Vihicle):
    def printName(self):
        print("Наземный вид транспорта")
    def printType(self):
        print("Автомобиль")
class Train(Vihicle):
    def printName(self):
        print("Наземный вид транспорта")
    def printType(self):
        print("Поезд")
car = Car()
car.printName()
car.printType()
train = Train()
train.printName()
train.printType()
```

Результат работы: Наземный вид транспорта
Автомобиль
Наземный вид транспорта
Поезд

Классы Car и Train являются наследниками от родительского класса Vihicle. Переопределив унаследованный метод printName, получилось, что два производных класса содержат метод с одним и тем же названием, но различным результатом работы. Это стало возможным благодаря полиморфизму. Таким образом, полиморфизм позволяет нам изменять функции в классах наследниках и переопределять методы.

Абстрагирование

Абстрагированием называется процесс выделения абстракций в предметной области задачи. Абстракция - это совокупность существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют особенности данного объекта с точки зрения дальнейшего рассмотрения и анализа. В соответствии с определением применяемая абстракция реального предмета существенно зависит от решаемой задачи: в одном случае нас будет интересовать форма предмета, в другом вес, в третьем - материалы, из которых он сделан, в четвертом – закон движения предмета и т.д.

Абстрактным называется класс, который содержит один и более абстрактных методов. Абстрактным называется объявленный, но не реализованный метод. Предполагается, что класс-потомок должен переопределить все методы и реализовать их функциональность.

В Python отсутствует встроенная поддержка абстрактных классов, для этой цели используется модуль abc (Abstract Base Class), который включается в стандартную библиотеку abc, добавляющую в язык абстрактные базовые классы. Модуль abc содержит метакласс (ABCMeta) и декораторы (@abstractmethod и @abstractproperty).

Рассмотрим пример создания абстрактного класса, который содержит один абстрактный и один обычный метод:

```
from abc import ABC, abstractmethod
class Vehicle(ABC):
    # обычный неабстрактный метод
    def reportChanges(self):
```

```
        print("Координаты машины были изменены")
    # абстрактный метод
    @abstractmethod
    def move(self):
        pass
```

Производный класс обязан переопределить и реализовать все методы или свойства, объявленные с использованием декораторов @abstractmethod и @abstractproperty, которые имеются в базовом абстрактном классе. Также следует учитывать, что если класс имеет хотя бы один абстрактный метод, то данный класс должен быть определен как абстрактный.

В приведённом выше примере также используется оператор pass, равноценный отсутствию операции. В ходе исполнения данного оператора ничего не происходит, поэтому он может использоваться в качестве заглушки в тех местах, где это синтаксически необходимо, например: в инструкциях, где тело является обязательным, таких как def, except и другие.

Зачастую pass используется там, где код пока ещё не появился, но планируется. Кроме этого, иногда, его используют при отладке, разместив на строчке с ним точку остановки.

Композиция

Особенностью объектно-ориентированного программирования является возможность реализовывать композиционный подход, заключающийся в том, что определяется класс-контейнер (агрегатор), который включает в себя вызовы других классов. В результате получается, что при создании объекта класса-контейнера, также создаются объекты включенных в него классов.

```
class WinDoor:
    def __init__(self, x, y):
        self.square = x * y
class Room:
    def __init__(self, x, y, z):
        self.square = 2 * z * (x + y)
        self.wd = []

    def addWD(self, w, h):
```

```
            self.wd.append(WinDoor(w, h))

        def workSurface(self):
            new_square = self.square
            for i in self.wd:
                new_square -= i.square
            return new_square


    r1 = Room(6, 3, 2.7)
    print(r1.square)  # выведет 48.6
    r1.addWD(1, 1)
    r1.addWD(1, 1)
    r1.addWD(1, 2)
    print(r1.workSurface())  # выведет 44.6
```

Таким образом, объект класс WinDoor создаётся при создании объекта класса Room.

ЗАДАЧИ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Рассмотрим выполнение лабораторной работы на примере нескольких задач.

Первая задача: создайте класс Car, содержащий метод location для изменения расстояния от машины до парковки, метод printLocation - для печати расстояния до парковки и метод speed, печатающий сообщение о том, что скорость автомобиля = 60 км/ч. Создайте два класса-наследника: Audi и BMW, для каждого задайте расстояние, на котором они находятся относительно парковки. Для автомобиля BMW переопределите метод speed, установив скорость = 70 км/ч. Сместите каждый из автомобилей на произвольное расстояние, используя метод location. Выведите информацию о скорости и удалённости от парковки каждого автомобиля.

Для начала работы необходимо создать вкладку в новом проекте (в качестве примера используется среда PyCharm). Теперь сделаем импорт необходимых библиотек и модулей.

```
from __future__ import print_function
import random
```

Псевдомодуль __future__ программисты могут использовать для включения новых языковых функций, которые не совместимы с текущим интерпретатором. В дальнейшем он будет необходим для изменения функции вывода сообщения. Модуль random, в свою очередь, позволяет генерировать случайные числа.

Теперь опишем поля и методы родительского класса Car:

```
class Car:
    loc = 0.0
    def __init__(self, loc):
        self.loc = loc
    def location (self):
        self.loc += (random.randint(30, 60))

    def speed (self):
```

```
        print("скорость автомобиля 60 км/ч")

    def printLocation(self):
        print(self.loc, end = ' ')
        print("км до парковки")
```

Обратим внимание, что при создании класса был явно указан конструктор, использующий ключевое слово self, с помощью которого были разграничены переменные класса (поля) и переменные метода, имеющие одинаковые называния.

Теперь создадим первый класс-наследник Audi. Данный класс будет наследовать методы и поля родительского класса, не переопределяя их и не добавляя новых. Так как в Python существует синтаксическое требование: блоки кода после if, except, def, class не могут быть пустыми, то используем оператор pass:

```
class Audi(Car):
        pass
```

Создадим второй класс-наследник BMW, который будет переопределять метод speed:

```
class BMW(Car):
    def speed (self):
        print("скорость автомобиля 70 км/ч")
```

Для завершения выполнения задания лабораторной работы создадим объекты классов Audi и BMW с определёнными параметрами и выведем данные о них:

```
car1=Audi(25)
car2=BMW(5)
print("Ауди ", end = ': ')#изменение функции в
команде вывода
car1.printLocation()
print("BMW", end = ': ')#изменение функции в команде
вывода
```

```
car2.printLocation()
print()
print("Состояние авто после перемещения по городу")
car2.location()
car1.location()
print("Ауди ", end = ': ')#изменение функции в
команде вывода
car1.printLocation()
car1.speed()
print("BMW", end = ': ')#изменение функции в команде
вывода
car2.printLocation()
car2.speed()
```

По умолчанию функция print после завершения вывода сообщения переносит позицию курсора для вывода на экран на новую строку, однако иногда необходимо, чтобы несколько последовательно вызываемых методов выводили сообщение на одну и ту же строку. Подключив псевдомодуль __future__, поставить запятую после перечисления необходимых переменных и прописать end = 'символ, которым будет заканчиваться вывод сообщения (\n по умолчанию) '.

Результат: Марка автомобиля - Audi
Расстояние до парковки равно 49,24
Скорость автомобиля равна 60 км/ч
Марка автомобиля - BMW
Скорость автомобиля равна 60 км/ч
Расстояние до парковки равно 5,00
Скорость автомобиля равна 70 км/ч

Вторая задача: написать программу с использованием абстрактного класса для конвертации валюты. Пользователь должен задать кол-во денег определённой валюты (USD или EU). Программа должна выводить результат конвертации на экран.

Так как по условию необходимо разработать программу с использованием абстрактного класса, то объявим абстрактный класс Valute, определим методы и подключим необходимые модули:

```
from __future__ import print_function
from abc import ABC, abstractmethod
class Valute(ABC):
    @abstractmethod
    def getRub(self):
        pass

    @abstractmethod
    def printData(self):
        pass
```

Создадим класс-наследник Convert, который будет переопределять и реализовывать все унаследованные абстрактные методы.

```
class Convert(Valute):
    Rub=0
    def __init__(self, USD, EU):
        self.USD = USD
        self.EU = EU
    def getRub(self):
        if (self.USD > 0):
            self.Rub = self.Rub + self.USD*65
        if (self.EU > 0):
            self.Rub = (self.Rub + self.EU*75)

    def printData(self):
        print("USD = ", +  self.USD)
        print("EU = ", + self.EU)
        print("При конвертации в рубли получим ",
end=' : ')
        print(self.Rub)
```

Таким образом, в классе Convert были реализованы описанные в интерфейсе методы, а также добавлен метод printData(). Для завершения выполнения задания лабораторной работы создадим объект convert класса Convert и вызовем для него последовательно необходимые методы. Стоит отметить, что при создании объекта необходимо указать количество валюты, которую необходимо перевести в рубли.

```
bag = Convert(3, 7)
bag.getRub()
bag.printData()
```

Результат: USD = 3
EU = 7
При конвертации в рубли получим : 720

Третья задача: написать программу, в которой создаётся массивы объектов таких классов, как: кальмар, дельфин, сом. Количество элементов каждого массива генерируется случайным образом от одного до четырёх.

Вывести на экран для каждого элемента массива тип созданной рыбы и её привычное место обитания (река/море/океан).

Для начала работы необходимо создать новую вкладку в проекте. Подключим необходимые бибилиотеки и опишем абстрактный класс Fish, который будет являться родительским:

```
import random
from abc import ABC, abstractmethod
class Fish(ABC):
    @abstractmethod
    def printData(self):
        pass
```

Создадим три класса-наследника, каждый из которых будет реализовывать объявленный абстрактный метод:

```
class Som(Fish):
    def printData(self):
        print("Сом, привычное место обитания - река")
class Dolphin(Fish):
    def printData(self):
        print("Дельфин, привычное место обитания -
море")
class Squid(Fish):
    def printData(self):
```

```
        print("Кальмар, привычное место обитания -
океан")
```

Создадим переменные, которые будут отвечать за количество элементов в классах Сом, Дельфин и Кальмар и присвоим им случайное значение от 1 до 4:

```
numS = (random.randint(1, 4))
numD = (random.randint(1, 4))
numSq = (random.randint(1, 4))
```

Для завершения выполнения задания лабораторной работы создадим массив объектов каждого класса и вызовем реализованные ранее методы:

```
print("Всего в заповеднике в заповеднике: ", +
(numS+numD+numSq), "животных")
som = [Som() for each in range(numS)]
for i in range (0,numS):
    som[i].printData()

dolphin = [Dolphin() for each in range(numD)]
for i in range (0,numD):
    dolphin[i].printData()

squid = [Squid() for each in range(numSq)]
for i in range (0,numD):
    squid[i].printData()
```

Результат: Всего в заповеднике в заповеднике: 6 животных
Сом, привычное место обитания - река
Сом, привычное место обитания - река
Дельфин, привычное место обитания - море
Дельфин, привычное место обитания - море
Кальмар, привычное место обитания - океан
Кальмар, привычное место обитания - океан

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Вариант 1

Задача 1. Создайте класс Robot, содержащий метод movement для изменения координат при движении(x,y), метод printPosition - для печати координат и метод color - для вывода сообщения, о том, что цвет робота белый. Создайте два класса-наследника: Wally и Eva, задайте их координаты. Переопределите метод color для класса Wally, заменив цвет на жёлтый. Сместите каждого робота на произвольное расстояние, используя movement. Выведите информацию о координатах и цвете каждого робота.

Задача 2. Необходимо создать два абстрактных класса, первый из которых описывает какой-либо объект (школьник, студент, аспирант) и реализуется в подклассах, второй - должен называться как суперкласс Pupil и также реализовываться в трёх подклассах.

Задача 3. В компьютерной игре Minecraft присутствуют овцы и прирученные волки. У овец может быть до 8.0 единиц здоровья, а у волков до 20.0. Положение волка или овцы задается трехмерным радиус-вектором от начала координат. У волка присутствует способность атаковать овцу: при атаке у овцы отнимается $(h / h_f)\ d / r^2$ единиц здоровья, где

- h_f — константа полного здоровья волка,
- h — текущее здоровье волка,
- d — константа урона волка, равная 4-м,
- r — расстояние от волка до овцы.

Необходимо создать модель атаки волка на овцу, позволяющую рассчитать, сколько секунд понадобится волку для убийства овцы при их заданных состояниях. Использовать абстрактный суперкласс.

Задача 4. Создать класс Human и описать в нем имя, пол, возраст и рост человека, при этом возраст и рост можно указывать как в формате целого числа, дробного и также в виде строки (57, пятьдесят семь; 180, 1.80, метр восемьдесят).Описать метод print, выводящий на экран всю информацию о человеке.

Вариант 2

Задача 1. Создайте класс Animals и два класса-наследника Home и Wild. Распределите животных ЛЕВ, МЕДВЕДЬ, СОБАКА, КОШКА по данным классам и добавьте породу/вид. Создайте для каждого класса-наследника метод getBreed для вывода породы/вида животного.

Задача 2. Написать программу с использованием абстрактного класса для перевода скорости. Пользователь должен ввести скорость спутника (предусмотреть ввод в км/ч и м/с). Программа должна выводить исходные данные и их перевод в км/с.

Задача 3. Создать класс Car и описать в нем марку машины, ее номер и скорость. Скорость можно указать как в формате целого числа, так и строки (60, шестьдесят). Описать метод Enter, заполняющий все поля класса с клавиатуры.

Задача 4. Сильная и независимая женщина решила исследовать процесс кормления своих питомцев на компьютерной модели. У каждого питомца есть параметр *сытость*, задаваемый целым числом процентов в диапазоне (0, 100]. При симуляции кормления (метод *.eat()*) модель собаки должна печатать в консоль "Гав!", а модель кошки "Мяу!". Кормление добавляет к сытости до 75% для любого животного, так, что в итоге сытость оказывается не более 100%. Необходимо реализовать такую модель, используя абстрактные классы, случайным образом сгенерировать массив питомцев и распечатать его состояние до и после кормления всех животных.

Вариант 3

Задача 1. Создайте класс Checkers, содержащий метод movement для изменения координат фигуры при перемещении (x,y), метод printPosition - для печати координат и метод color - для вывода сообщения, о том, что цвет фигуры белый. Создайте класс-наследник Queen, начальные координаты которого (0,0), переопределите метод printPosition таким образом, чтобы при выходе за границы игрального поля выводилось сообщение о невозможности хода. Создайте класс-наследник Pawn (0,0) класса Queen, переопределите метод color, изменив цвет фигуры на чёрный. Сместите фигуры на произвольное расстояние.Выведите информацию о координатах и цвете каждой из фигур

Задача 2. Написать программу с использованием абстрактного класса для вычисления синуса угла, заданного рядом Тейлора.

Задача 3. Создать класс MyWallet и описать в нем имя владельца счёта, номер карты и остаток денег на счёте, при этом остаток денег на счёте можно указывать как в формате дробного числа, так и строки (1000.00, тысяча рублей). Описать метод rest, определяющий, есть ли на счету деньги, и если есть, то сколько.

Задача 4. В очереди за бесплатными макбуками стоят студенты групп ИТД и ЭВМ. У каждого студента есть целочисленный балльный рейтинг, изначально равный нулю. Студенты могут сдавать лабораторные работы по программированию (метод *.makeProgramming()*) и по электротехнике (метод *.makeElecEngineering()*). За работу по программированию студенту ЭВМ зачисляется случайным образом 3 или 4 балла, а за электротехнику 4 или 5. Для студентов ИТД наоборот. Необходимо случайным образом сгенерировать модель очереди студентов, провести

несколько работ, и определить, сколько студентов ИТД и сколько студентов ЭВМ получат макбук. Макбуков раздается около 25% от количества всех студентов, и получают из лучшие по рейтингу. После каждого существенного шага следует выводить на экран состояния студентов, для реализации модели необходимо использовать абстрактные классы и оператор *instanceof*.

Вариант 4

Задача 1. Создайте класс Geolocation, содержащий метод forward для изменения координат при движении по оси Ox и Oy, метод decline -для изменения координат по осям Ox, Oy,Oz, метод printPosition – для печати координат. Создайте класс-наследник aircraft и определите на нём метод wing, печатающий сообщение о том, что у самолёта 2 крыла. Создайте класс-наследник aircraft, переопределяющий метод wing. Сместите самолёт и вертолёт на произвольные расстояния и выведите сообщение о количестве крыльев и координатах летательных аппаратов.

Задача 2. Создать 3 класса (базовый) и 2 предка, которые описывают некоторых работников с почасовой оплатой (один из предков) и фиксированной оплатой(второй предок).Описать в базовом классе абстрактный метод для расчета среднемесячной зарплаты. Для «почасовиков» формула для расчета такая: “среднемесячная зарплата=20.8*8*ставка в час”, для работников с фиксированной оплатой “среднемесячная зарплата = фиксированной месячной оплате”. Задание: упорядочить всю последовательность рабочих по убыванию среднемесячной зарплаты. При совпадении зарплаты – упорядочить данные в алфавитном порядке по имени. Вывести идентификатор работника, имя и среднемесячную зарплату для всех элементов списка.

Задача 3. Создать класс Mail и описать в нем имя отправителя, имя получателя, место получения и место отправления, при этом место

отправления и получения могут быть заданы индексами(целым числом), либо названием города(строкой). Описать метод travel, определяющий покинет ли письмо пределы города (места получения и отправления отличаются).

Задача 4. Требуется реализовать часть виртуальной машины языка Brainfuck: список из объектов-операций, имеющих метод .execute(), принимающий по ссылке массив памяти и номер активной ячейки. Операции в BF:

1. '>' — перейти к следующей ячейке;
2. '<' — перейти к предыдущей ячейке;
3. '+' — увеличить значение в текущей ячейке на 1;
4. '-' — уменьшить значение в текущей ячейке на 1;
5. '.' — напечатать значение из текущей ячейки,
6. ',' — ввести значение и сохранить в текущей ячейке;
7. '[' — если значение текущей ячейки ноль, перейти вперёд по тексту программы на ячейку, следующую за соответствующей ']' (с учётом вложенности);
8. ']' — если значение текущей ячейки не нуль, перейти назад по тексту программы на символ '[' (с учётом вложенности).

Применить абстрактный класс для задания операции.

Вариант 5

Задача 1. Создайте класс Parent, задайте параметр grow определите метод colorEye, выводящий предложение о зелёном цвете глаз, метод changeGrow – вычисляющий изменение роста, метод printGrow – печатающий величину роста. Создайте два класса-наследника Masha и Peter, задайте параметр роста для них. В одном из данных классов переопределите метод colorEye, заменив цвет глаз на карий. Задайте параметры изменения роста для имеющихся классов. Выведите сообщение о росте и цвете глаз каждого класса.

Задача 2. Создать абстрактный класс, описывающие возможности игровых персонажей (бег, плавание, прыжки). Создать базовый класс персонажа для игры и два унаследованных класса персонажа, которые используют реализованные методы абстрактного класса для определения своих возможностей.

Задача 3. Создать класс Holiday и описать в нем название праздника, имя приглашенного и количество персон в этом приглашении. Описать метод family, определяющий может ли вся семья приглашенного (количество задается с клавиатуры) посетить данный праздник.

Задача 4. Для описания всех людей, находящихся в магазине, необходимо выделить подмножество работников и подмножество посетителей. У каждого человека есть полное имя, у сотрудников магазина имеет значение должность, а у посетителей возраст. Необходимо сгенерировать список людей в магазине и вызовом виртуального метода из абстрактного класса напечатать для сотрудников фамилию и должность, а для посетителей имя и возраст.

Вариант 6

Задача 1. Создайте класс Factory, задайте параметры координат x, y. Определите метод geolocation, печатающий координаты месторасположения. Создайте класс-наследник Employee, задайте его координаты. Создайте для него метод ChangePosition для изменения координат при движении, метод department – для вывода сообщения о принадлежности работника к отделу продаж. Сместите Employee на произвольное расстояние. Выведите сообщение о координатах завода, координатах рабочего и отделе, где он работает, а также - о координатах расстояния между заводом и рабочим.

Задача 2. Реализовать при помощи абстрактного класса подсчет факториала числа n и двойного факториала числа n, выполнить проверку и вывести результат на экран.

Задача 3. Создать класс Donor и описать в нем имя реципиента, его группу крови, имя донора и его группу крови, группа крови может задаваться как строкой, так и целым числом. Описать метод blood, определяющий может ли человек быть донором для данного больного (одинаковая группа крови).

Задача 4. При генерации выписки реквизитов клиентов оператора фискальных данных для юридических лиц печатается название, ИНН и ОГРН, а для индивидуальных предпринимателей фамилия, ИНН и ОГРНИП. Требуется сгенерировать список клиентов ОФД и напечатать выписки реквизитов, используя виртуальный метод из абстрактного класса.

Вариант 7

Задача 1. Создайте класс Flower, определите метод color, выводящий сообщение о цвете цветка, метод components, подсчитывающий число листков и лепестков в цветке и метод printCompononts, выводящий на экран число компонентов цветка. Создайте два класса наследника: Rose and Tulip. Задайте для них переменные, отвечающие за количество лепестков и цвет, а для класса Rose задайте ещё параметр количества шипов. Переопределите для класса Pose метод components, добавив в расчёты параметр количества шипов. Выведите для каждого цветка его цвет и количество компонентов.

Задача 2. Создать абстрактного класса Pair с виртуальными арифметическими операциями(+-*/) и реализовать произвольный класс Rational (класс рациональных чисел) и на примере дробей

протестировать все арифметические операции. Вывести все операции и полученный результат на экран.

Задача 3. Создать класс Birthday и описать в нем имя человека, день его рождения, месяц и год, при этом месяц может задаваться, как номером(целым числом), так и словом(строкой). Описать метод print, выводящий дату рождения в формате dd.mm.yyyy.

Задача 4. Требуется рассчитать периметр и площадь фигур различной геометрической формы. Создать абстрактный класс Shape и реализовать в нем абстрактные методы, которые должны вычислять периметр и площадь заданной геометрической формы. Для данной задачи рассчитайте площадь квадрата, круга и прямоугольного треугольника.

Вариант 8

Задача 1. . Создайте класс Anima, задайте для него метод location, выводящий сообщение о том, что животное живёт в зоопарке, метод amountOfFeed, входная переменная которого отвечает за количество животных одного типа, а сам метод рассчитывает количество необходимого корма. Также создайте метод printAmountOfFeed, выводящий сообщение о количестве корма, который необходим животным. Создайте два класса наследника: Penguin и Giraffe. Введите и количество. Переопределите метод location для класса Penguin, изменив его место жительства на Арктику. Выведите для каждого типа животных их место жительства и необходимое количество корма для их питания.

Задача 2. Написать программу с использованием абстрактного класса для определения средней оценки студента. Пользователь должен все текущие оценки и наименование предмета. Подсчитать предполагаемый средний балл и вывести результат на экран.

Задача 3. Создать класс Navigator и описать в нем номер автомобиля, место отправления и место назначения (можно задать как номером региона, так и названием города).

Описать метод travel, определяющий, покинет ли машина данный регион.

Задача 4. Требуется рассчитать объем фигур различной геометрической формы. Создать абстрактный класс Shape и реализовать в нём абстрактные методы, которые должны вычислять объем заданного тела вращения. Для данной задачи рассчитайте объем куба, шара и цилиндра.

Вариант 9

Задача 1. Создайте класс Student, определите метод amountOfHomework, подсчитываюший сумму необходимых к выполнению домашних заданий по трём предметам, получившееся число метод printAmountOfStudent выводит на экран; метод group, выводит на экран сообщение о том, что студент обучается в группе ИТД. Создайте два класса-наследника Kate и Peter. Для класса Peter переопределите метод amountOfHomework, убавив один входной параметр. Введите количество заданий по предметам, необходимых для выполнения каждым студентом. Выведите для каждого студента сумму необходимых к выполнению домашних заданий и его группу.

Задача 2. Создать класс, описывающий структуру комплексного числа. Используя абстрактный класс, найти и вывести на экран сопряженное ему число.

Задача 3. Создать класс Book и описать в нем имя автора книги и ее идентификация в библиотеке. Идентификация может быть кодом книги (8-значное число) или названием (строка).

Описать метод order, определяющий, должна ли книга стоять до или после данной (по номеру или по алфавиту)

Задача 4. Создать абстрактный базовый класс «Товар» с методами, которые:

- Выводят на экран информацию о товаре (Наименование, срок годности)
- Определяют, соответствует ли данный товар сроку годности на данный момент*

Сформировать массив из n товаров, вывести полную информацию о товарах из массива на экран, а также найти просроченный товар (на момент текущей даты).

* - для данной задачи используйте класс GregorianCalendar из библиотеки Java.

Вариант 10

Задача 1. Создайте класс Peoples, задайте ФИО. Создайте класс-наследник Students, заполните его ФИО. Создайте для него метод ChangeAstimate для изменения оценок студента, метод WriteAstimate – для вывода массива с оценками и ФИО. Задайте произвольные ФИО и оценки. Выведите список, состоящий из трёх студентов.

Задача 2. Написать программу с использованием абстрактного класса для нахождения среднего арифметического заданного ряда целых чисел.

Задача 3. Создать класс Password и описать в нем ник человека на сайте и его пароль, при этом пароль может быть четырёхзначным пин-кодом (целое число, состоящее из 4 цифр) или полноценным паролем (строкой). Описать метод coincedence, определяющий, является ли введённое с клавиатуры n паролем для данного ника.

Задача 4. Создать абстрактный базовый класс «Triangle» для треугольника и вычислить площадь и периметр с помощью абстрактных методов. Поля данных должны включать две стороны и угол между ними. Определить производные классы: прямоугольный треугольник, равнобедренный треугольник, равносторонний треугольник с собственными функциями вычисления площади и периметра.

Вариант 11

Задача 1. Создайте класс Animals и два класса-наследника Home и Wild. Распределите животных ЛЕВ, МЕДВЕДЬ, СОБАКА, КОШКА по данным классам и добавьте породу/вид. Создайте для каждого класса-наследника метод GetBreed для вывода породы/вида животного.

Задача 2. Создать класс для описания понятия «вершина», содержащий в себе название горной вершины и ее высоту. С помощью абстрактного класса определить самую высокую и самую низкую вершины.

Задача 3. Создать класс Cards и описать в нем масть и значение карты(целое число или строка).

Описать метод определяющий, побьет ли эта карта девятку. (масть не учитывать)

Задача 4. Требуется рассчитать периметр и площадь фигур различной геометрической формы. Создать абстрактный класс Shape и реализовать в нем абстрактные методы, которые должны вычислять периметр и площадь заданной геометрической формы. Для данной задачи рассчитайте площадь квадрата, круга и прямоугольного треугольника.

Вариант 12

Задача 1. Создайте класс SunSystem и два класса наследника Planets и Satellites для которых определите диаметр и привязку спутника к планете. Опишите метод вывода планеты на экран GetPlanet. При выводе на экран планеты автоматически выводится ее диаметр и ее спутники.

Задача 2. Дан класс «Библиотека», содержащий информацию: название книги, автора и год издания. Используя абстрактный класс, определить названия книги и авторов, изданных в 1968 году.

Задача 3. Создать класс Student и описать в нем имя ученика, оценки по математике, физике и информатике. При этом оценки могут быть в формате числа(4) или строки(четыре). Описать метод mark, выводящий на экран, является ли ученик отличником, хорошистом или ни тем, ни другим.

Задача 4. Требуется рассчитать объем фигур различной геометрической формы. Создать абстрактный класс Shape и реализовать в нём абстрактные методы, которые должны вычислять объем заданного тела вращения. Для данной задачи рассчитайте объем куба, шара и цилиндра.

Вариант 13

Задача 1. Создайте класс Pond и два класса наследника Sea и Lake. Для каждого задайте размер водоёма и для класса Sea минимальный размер. На ввод с клавиатуры приходит название водоёма и его размер – в ответе программы содержится информация о типе водоёма (Sea или Lake соответственно).

Задача 2. Создать класс для описания рационального дробного числа, содержащий объекты ЧИСЛИТЕЛЬ и ЗНАМЕНАТЕЛЬ. Создать абстрактный класс, позволяющий определить равенство чисел.

Задача 3. Создать класс Mail и описать в нем имя получателя, количество писем и почтовое отделение. Почтовое отделение может быть задано номером или названием. Описать метод destination, определяющий, находится ли клиент в нужном отделении по введенному отделению.

Задача 4. В очереди за бесплатными макбуками стоят студенты групп ИТД и ЭВМ. У каждого студента есть целочисленный балльный рейтинг, изначально равный нулю. Студенты могут сдавать лабораторные работы по программированию (метод *.makeProgramming()*) и по электротехнике (метод *.makeElecEngineering()*). За работу по программированию студенту ЭВМ зачисляется случайным образом 3 или 4 балла, а за электротехнику 4 или 5. Для студентов ИТД наоборот. Необходимо случайным образом сгенерировать модель очереди студентов, провести несколько работ, и определить, сколько студентов ИТД и сколько студентов ЭВМ получат макбук. Макбуков раздается около 25% от количества всех студентов, и получают из лучшие по рейтингу. После каждого существенного шага следует выводить на экран состояния студентов, для реализации модели необходимо использовать абстрактные классы и оператор *instanceof*.

Вариант 14

Задача 1. Создайте класс Staff и два класса наследника Manager и Cashier. Определите график работы по дням недели для каждого из классов. Создайте метод GetWork для вывода графика работы каждого класса и метод TodayWork, принимающий на вход числовое значение дня недели и отвечающий работает ли сегодня персонал или нет.

Задача 2. Создать класс АВТОМОБИЛИСТЫ, содержащий не более 10 объектов, указав фамилию, марку (модель) автомобиля и его номер. Используя абстрактный класс поиска, вывести на экран фамилию владельца и номер машины конкретной марки.

Задача 3. Создать класс Dream и описать в нем имя человека и количество часов сна за последние сутки(восемь или 8). Описать метод normal, определяющий выспался ли человек(проспал более 8 часов) и если нет, то сколько часов сна ему еще нужно.

Задача 4. В компьютерной игре Minecraft присутствуют овцы и прирученные волки. У овец может быть до 8.0 единиц здоровья, а у волков до 20.0. Положение волка или овцы задается трехмерным радиус-вектором от начала координат. У волка присутствует способность атаковать овцу: при атаке у овцы отнимается $(h / h_f)\, d / r^2$ единиц здоровья, где

- h_f — константа полного здоровья волка,
- h — текущее здоровье волка,
- d — константа урона волка, равная 4-м,
- r — расстояние от волка до овцы.

Необходимо создать модель атаки волка на овцу, позволяющую рассчитать, сколько секунд понадобится волку для убийства овцы при их заданных состояниях. Использовать абстрактный суперкласс.

.

Вариант 15

Задача 1. Создайте класс Rooms и определите для него кол-во комнат в номере гостиницы. Распределите номера (от 1-го до 10-го) по кол-ву комнат в нём, создав классы наследники. Опишите метод GetRoom, получающий на вход число человек в семье, заезжающей в номер и выводящий число подходящего для семьи номера из расчёта 1 комната на одного человека.

Задача 2. Необходимо создать абстрактный класс, использующийся для вывода на экран информации о неуспевающих студентах (фамилия и группа). Информация о студентах находится в классе СТУДЕНТ: фамилия, группа, оценка за экзамен.

Задача 3. Создать класс Student и описать в нем 2 оценки за экзамен (число или строка) и 2 зачета(+ или -). Описать метод scholarship, определяющий, получит ли человек стипендию (оценки за экзамен не ниже 4 и получены зачеты).

Задача 4. Создать абстрактный базовый класс «Товар» с методами, которые:

- Выводят на экран информацию о товаре (Наименование, срок годности)
- Определяют, соответствует ли данный товар сроку годности на данный момент*

Сформировать массив из n товаров, вывести полную информацию о товарах из массива на экран,

а также найти просроченный товар (на момент текущей даты).

* - для данной задачи используйте класс GregorianCalendar из библиотеки Java.

Вариант 16

Задача 1. Создайте класс Checkers, содержащий метод movement для изменения координат фигуры при перемещении (x,y), метод printPosition - для печати координат и метод color - для вывода сообщения, о том, что цвет фигуры белый. Создайте класс-наследник Queen, начальные координаты которого (0,0), переопределите метод printPosition таким образом, чтобы при выходе за границы игрального поля выводилось сообщение о невозможности хода. Создайте класс-наследник Pawn (0,0) класса Queen, переопределите метод color,

изменив цвет фигуры на чёрный. Сместите фигуры на произвольное расстояние. Выведите информацию о координатах и цвете каждой из фигур

Задача 2. Дан класс ЛЮДИ: пол, год рождения, цвет глаз. Используя абстрактный класс, вывести на экран количество кареглазых женщин 1980 года рождения.

Задача 3. Создать класс Mail и описать в нем имя отправителя, имя получателя, место получения и место отправления, при этом место отправления и получения могут быть заданы индексами(целым числом), либо названием города(строкой).

Описать метод travel, определяющий покинет ли письмо пределы города (места получения и отправления отличаются).

Задача 4. Создать абстрактный базовый класс «Triangle» для треугольника и вычислить площадь и периметр с помощью абстрактных методов. Поля данных должны включать две стороны и угол между ними. Определить производные классы: прямоугольный треугольник, равнобедренный треугольник, равносторонний треугольник с собственными функциями вычисления площади и периметра.

Вариант 17

Задача 1. Создайте класс Student, определите метод amountOfHomework, подсчитываюший сумму необходимых к выполнению домашних заданий по трём предметам, получившееся число метод printAmountOfStudent выводит на экран; метод group, выводит на экран сообщение о том, что студент обучается в группе ИТД. Создайте два класса-наследника Kate и Peter. Для класса Peter переопределите метод amountOfHomework, убавив один входной параметр. Введите количество заданий по предметам, необходимых для

выполнения каждым студентом. Выведите для каждого студента сумму необходимых к выполнению домашних заданий и его группу.

Задача 2. Написать программу с использованием абстрактного класса для перевода чисел в двоичную систему счисления. Пользователь должен ввести десятичное число, а получить результат на экране в двоичном виде.

Задача 3. Создать класс Book и описать в нем имя автора книги и ее идентификация в библиотеке. Идентификация может быть кодом книги (8-значное число) или названием (строка).

Описать метод order, определяющий, должна ли книга стоять до или после данной (по номеру или по алфавиту)

Задача 4. Для описания всех людей, находящихся в магазине, необходимо выделить подмножество работников и подмножество посетителей. У каждого человека есть полное имя, у сотрудников магазина имеет значение должность, а у посетителей возраст. Необходимо сгенерировать список людей в магазине и вызовом виртуального метода из абстрактного класса напечатать для сотрудников фамилию и должность, а для посетителей имя и возраст.

Вариант 18

Задача 1. Создайте класс SunSystem и два класса наследника Planets и Satellites для которых определите диаметр и привязку спутника к планете. Опишите метод вывода планеты на экран GetPlanet. При выводе на экран планеты автоматически выводится ее диаметр и ее спутники.

Задача 2. Реализовать при помощи абстрактного класса поиск чисел Фибоначчи из заданного ряда целых чисел.

Задача 3. Создать класс Car и описать в нем марку машины, ее номер и скорость. Скорость можно указать как в формате целого числа, так и строки (60, шестьдесят). Описать метод enter, заполняющий все поля класса с клавиатуры.

Задача 4. Сильная и независимая женщина решила исследовать процесс кормления своих питомцев на компьютерной модели. У каждого питомца есть параметр *сытость*, задаваемый целым числом процентов в диапазоне (0, 100]. При симуляции кормления (метод *.eat()*) модель собаки должна печатать в консоль "Гав!", а модель кошки "Мяу!". Кормление добавляет к сытости до 75% для любого животного, так, что в итоге сытость оказывается не более 100%. Необходимо реализовать такую модель, используя абстрактные классы, случайным образом сгенерировать массив питомцев и распечатать его состояние до и после кормления всех животных.

Вариант 19

Задача 1. Создайте класс Staff и два класса наследника Manager и Cashier. Определите график работы по дням недели для каждого из классов. Создайте метод GetWork для вывода графика работы каждого класса и метод TodayWork, принимающий на вход числовое значение дня недели и отвечающий работает ли сегодня персонал или нет.

Задача 2. Написать программу с использованием абстрактного класса для вычисления синуса угла, заданного рядом Тейлора.

Задача 3. Создать класс Mail и описать в нем имя получателя, количество писем и почтовое отделение. Почтовое отделение может быть задано номером или названием.

Описать метод destination, определяющий, находится ли клиент в нужном отделении по введенному отделению.

Задача 4. Требуется реализовать часть виртуальной машины языка Brainfuck: список из объектов-операций, имеющих метод .execute(), принимающий по ссылке массив памяти и номер активной ячейки. Операции в BF:

1. '>' — перейти к следующей ячейке;
2. '<' — перейти к предыдущей ячейке;
3. '+' — увеличить значение в текущей ячейке на 1;
4. '-' — уменьшить значение в текущей ячейке на 1;
5. '.' — напечатать значение из текущей ячейки,
6. ',' — ввести значение и сохранить в текущей ячейке;
7. '[' — если значение текущей ячейки ноль, перейти вперёд по тексту программы на ячейку, следующую за соответствующей ']' (с учётом вложенности);
8. ']' — если значение текущей ячейки не нуль, перейти назад по тексту программы на символ '[' (с учётом вложенности).

Применить абстрактный класс для задания операции.

Вариант 20

Задача 1. Создайте класс Flower, определите метод color, выводящий сообщение о цвете цветка, метод components, подсчитывающий число листков и лепестков в цветке и метод printCompononts, выводящий на экран число компонентов цветка. Создайте два класса наследника: Rose and Tulip. Задайте для них переменные, отвечающие за количество лепестков и цвет, а для класса Rose задайте ещё параметр количества шипов. Переопределите для класса Pose метод components, добавив в расчёты параметр количества шипов. Выведите для каждого цветка его цвет и количество компонентов.

Задача 2. Дан класс «Библиотека», содержащий информацию: название книги, автора и год издания. Используя абстрактный класс, определить названия книги и авторов, изданных в 1968 году.

Задача 3. Создать класс Password и описать в нем ник человека на сайте и его пароль, при этом пароль может быть четырёхзначным пин-кодом (целое число, состоящее из 4 цифр) или полноценным паролем (строкой). Описать метод coincedence, определяющий, является ли введённое с клавиатуры n паролем для данного ника.

Задача 4. Требуется рассчитать объем фигур различной геометрической формы. Создать абстрактный класс Shape и реализовать в нём абстрактные методы, которые должны вычислять объем заданного тела вращения. Для данной задачи рассчитайте объем куба, шара и цилиндра.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка заданий.
4. UML-диаграммы классов, используемых в программе, для каждого задания.
5. Листинг программы для каждого задания.
6. Результаты выполнения программ.
7. Выводы по работе в целом.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение термину объектно-ориентированное программирование.
2. Раскройте значения следующих терминов: класс, объект.
3. Опишите назначение и процесс вызова явно и неявно определённых конструкторов и деструкторов.
4. Запишите фрагмент кода программы на языке Java для создания класса Car с методом printLocation и создайте объект данного класса.
5. Изобразите схематично многоуровневую иерархию наследования.
6. Опишите специфику принципа полиморфизм.
7. Расшифруйте обозначение abc и обоснуйте необходимость использования данного модуля.
8. Приведите пример использования ключевого слова this при создании метода.
9. Раскройте необходимость ограничения доступа к данным.
10. Объясните функцию запрета наследования.
11. Опишите преимущества использования абстрактных классов.
12. Запишите фрагмент кода программы на языке Java для перебора всех элементов массива и подсчета их среднего арифметического.
13. Опишите назначение каждого из четырёх основных принципов ООП.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Разработать программы согласно варианту.
4. Выполнить тестирование программ.
5. Продемонстрировать работу программ преподавателю.
6. Оформить отчет.
7. Защитить выполненную работу у преподавателя.

ЛАБОРАТОРНАЯ РАБОТА №4
ДОПОЛНИТЕЛЬНЫЕ БИБЛИОТЕКИ ЯЗЫКА PYTHON

Целью выполнения лабораторной работы является формирование практических навыков процедурного программирования, разработки и отладки программ, овладение методами и средствами разработки и оформления технической документации.

Основными задачами выполнения лабораторной работы являются:

1. Научиться загружать дополнительные библиотеки в среду разработки;
2. Изучить особенности и возможности библиотек NumPy, SciPy, Matplotlib и Pillow;
3. Изучить типовые алгоритмы решения задач с использованием дополнительных библиотек.

Результатами работы являются:

1. Реализация разработанных алгоритмов на языке программирования Python;
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Библиотека — это сборник подпрограмм или объектов, используемых для разработки программного обеспечения (ПО). Она может включать данные конфигурации, документацию, справочные данные, шаблоны сообщений, предварительно написанный код. В языке Python библиотеки представляют собой модуль, то есть функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом, предназначенный для использования в других программах. Они необходимы для упрощения программирования: становится возможной реализация необходимой операции без написания излишнего кода.

В рамках данной лабораторной работы речь пойдёт о четырёх популярных библиотеках: NumPy, SciPy, Matplotlib и Pillow. Для того, чтобы использовать их при реализации проектов, необходимо их подключить. Подключение библиотек будем рассматривать на примере программы PyCharm.

Для начала необходимо во вкладке “File” выбрать пункт “Settings” (см.рис.1):

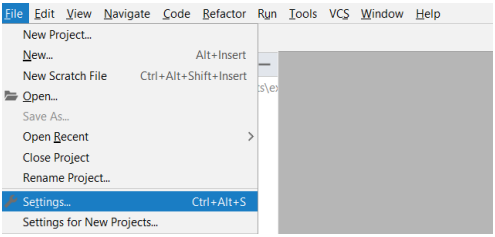


Рис.1. Настройка среды

Затем выберите пункт “Project Interpreted”, расположенный во вкладке “Project”. Откроется окно, в котором будут перечислены все установленные библиотеки. Для добавления новой необходимо нажать

на “+” (см. рис.2) и написать название необходимой библиотеки, она будет установлена в среду.

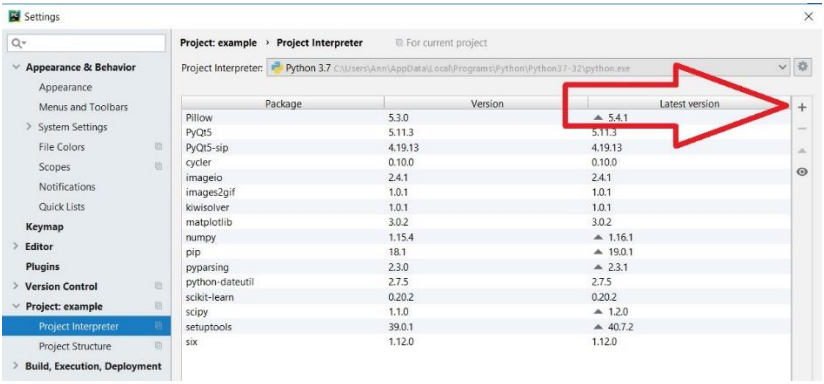


Рис.2. Добавление новой библиотеки

Библиотека NumPy

NumPy (сокращение от Numerical Python — “числовой Python”) обеспечивает эффективный интерфейс для хранения и работы с плотными буферами данных. Массивы библиотеки NumPy похожи на встроенный тип данных языка Python list, но обеспечивают гораздо более эффективное хранение и операции с данными при росте размера массивов. Они также являются базовыми для большинства практически всей экосистемы утилит для исследования данных Python.

Для подключения библиотеки к текущей рабочей вкладке необходимо сделать её импорт:

```
import numpy as np
```

Данная синтаксическая конструкция означает, что в коде программы к данным библиотеки NumPy будем обращаться, используя не полное название numpy, а псевдоним np. Данное решение делает код мобильнее и читабельнее.

Для печати версии и конфигурации необходимо импортировать библиотеку и прописать:

```
print(np.__version__)
np.show_config()
```

Результат: 1.15.4
blas_mkl_info:
NOT AVAILABLE
blis_info:
…

Для работы с массивами посредством использования библиотеки NumPy применяется большое количество операций, однако их все можно подразделить на несколько основных типов:

1. создание массивов;
2. получение атрибутов массивов;
3. индексацию массивов;
4. операции с массивами.

Создание массивов

Библиотека NumPy позволяет создавать матрицы и массивы, которые уже заполнены определёнными числами и цифрами.

Рассмотрим несколько примеров:

1. Создание массива, заполненного 10 нулями:

```
Z = np.zeros(10)
print(Z)#[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

2. Создание массива, заполненного 10 единицами:

```
Z = np.ones(10)
print(Z)#[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3. Создание массива, заполненного 5 цифрами 2,5:

```
Z = np.full(10, 2.5)
print(Z) #[2.5 2.5 2.5 2.5 2.5]
```

4. Создание заполненного числами от 11 до 16 одномерного массива:

```
Z = np.arange(11,17)
print(Z)#[11 12 13 14 15 16]
```

5. Создание матрицы размерности 3x3, заполненной цифрами от 0 до 8:

```
Z = np.arange(9).reshape(3,3)
print(Z) #[[0 1 2][3 4 5] [6 7 8]]
```

6. Создание единичной матрицы размерности 3x3:

```
Z = np.eye(3)
print(Z)#[[1. 0. 0.] [0. 1. 0.] [0. 0. 1.]]
```

7. Создать 4x4 матрицу и заполнить её в шахматном порядке, используя функцию tile:

```
Z = np.tile(np.array([[0,1],[1,0]]),(2,2))
print(Z)#[[0 1 0 1][1 0 1 0][0 1 0 1][1 0 1 0]]
```

8. Создать 4x4 матрицу и заполнить её в шахматном порядке, не используя функцию tile:

```
Z = np.zeros((3,3), dtype=int)
Z[1::2,::2] = 1
Z[::2,1::2] = 1
print(Z)#[[0 1 0][1 0 1][0 1 0]]
```

9. Создание матрицы с 0 внутри, и 1 на границах:

```
Z = np.ones((4,4))
Z[1:-1,1:-1] = 0
print(Z)
```

Результат: [[1. 1. 1. 1.]
[1. 0. 0. 1.]
[1. 0. 0. 1.]
[1. 1. 1. 1.]]

10. Создать 3x3 матрицу со значениями в строках от 0 до 2:

```
Z = np.zeros((3,3))
Z += np.arange(3)
print(Z) #[[0. 1. 2.][0. 1. 2.][0. 1. 2.]]
```

Атрибуты массивов

Каждый массив содержит определённые атрибуты, знание которых зачастую необходимо для продолжения работы над проектом. Приведём пример получения некоторых из них:

1. Размерность массива:

```
# трехмерный массив
x3 = np.random.randint(10, size=(3, 4, 5))
print("Разменость матрицы: ", x3.ndim)#3
```

2. Размерность каждого измерения массива:

```
# трехмерный массив
x3 = np.random.randint(10, size=(3, 4, 5))
print("x3 shape:", x3.shape)
```

3. Общий размер массива:

```
# трехмерный массив
x3 = np.random.randint(10, size=(3, 4, 5))
print("x3 size: ", x3.size)#x3 size:  60
```

4. Тип данных элементов массива:

```
# трехмерный массив
x3 = np.random.randint(10, size=(3, 4, 5))
print("dtype:", x3.dtype)#dtype: int32
```

5. Размерность в байтах каждого элемента массива:

```
# трехмерный массив
x3 = np.random.randint(10, size=(3, 4, 5))
print("itemsize:", x3.itemsize,
"bytes")#itemsize: 4 bytes
```

6. Полный размер массива в байтах

```
# трехмерный массив
x3 = np.random.randint(10, size=(3, 4, 5))
print("nbytes:", x3.nbytes, "bytes")#nbytes:
240 bytes
```

Индексация массивов

Для того, чтобы производить с массивами различного типа операции, зачастую необходимо обращаться конкретно к необходимому элементу массива. В Python нумерация элементов массива начинается с 0.

Приведём пример различных способов обращений к элементам массива:

1. Обращение к первому элементу с начала:

```
x1= np.random.randint(10, size=6)
print(x1[0])
```

2. Обращение к последнему элементу массива:

```
np.random.randint(10, size=6)
print(x1[-1])
```

3. Обращение к первому элементу двумерного массива:

```
np.random.randint(10, size=(3, 4))
print( x2[0, 0])
```

4. Изменение значения первого элемента двумерного массива:

```
np.random.randint(10, size=(3, 4))
x2[0, 0] = 12
```

5. “Разворот” массива:

```
Z = np.arange(10)
print (Z)#[0 1 2 3 4 5 6 7 8 9]
Z = Z[::-1]
print(Z)#[9 8 7 6 5 4 3 2 1 0]
```

6. Поиск индексов ненулевых элементов массива:

```
nz = np.nonzero([1,2,0,0,4,0])
print(nz)#(array([0, 1, 4], dtype=int32),)
```

Операции с массивами

Библиотека NumPy содержит набор операций, которые значительно упрощают реализацию различных функций при работе с массивами, приведём основные из них:

1. Поиск максимального и минимального элемента:

```
import numpy as np
Z = np.random.random(4)
print(Z)
m = Z.mean()
print(m)
```

Результат: [0.36984785 0.07246234 0.76453851 0.17051021]
0.3443397287490455

2. Поиск среднего значения элементов массива:

```
import numpy as np
Z = np.random.random(4)
print(Z)
m = Z.mean()
print(m)
```

Результат: [0.02368754 0.93823604 0.05621458 0.67308608]
0.4228060575152336

3. Поиск индекса 100 элемента массива размерности (6,7,8):

```
import numpy as np
print(np.unravel_index(100, (6,7,8)))
```

Результат: (1, 5, 4)

4. Перемножение единичных матриц 2x3 и 3x1:

```
import numpy as np
Z = np.dot(np.ones((2,3)), np.ones((3,1)))
print(Z)
```

Результат: [[3.]
[3.]]

5. Замена знака элементов массива, значения которых находятся в промежутке (3;8]:

```
import numpy as np
Z = np.arange(11)
print(Z)
Z[(3 < Z) & (Z <= 8)] *= -1
print(Z)
```

Результат: [0 1 2 3 4 5 6 7 8 9 10]
[0 1 2 3 -4 -5 -6 -7 -8 9 10]

6. Сортировка массива по возрастанию:

```
import numpy as np
Z = np.arange(10)
Z[2]=19
print(Z)
Z.sort()
print(Z)
```

Результат: [0 1 19 3 4 5 6 7 8 9]
[0 1 3 4 5 6 7 8 9 19]

7. Поиск элемента массива, ближайшего к заданному значению числа:

```
import numpy as np
Z = np.arange(10)
print(Z)
v = np.random.uniform(0,100)
print(v)
index = (np.abs(Z-v)).argmin()
print(Z[index])
```

Результат: [0 1 2 3 4 5 6 7 8 9]
2.6004062844243947
3

8. Преобразование элементов массива из int в float:

```
import numpy as np
Z = np.arange(10, dtype=np.int32)
print(Z)
Z = Z.astype(np.float32, copy=False)
print(Z)
```

Результат: [0 1 2 3 4 5 6 7 8 9]
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]

9. Изменение положения строк в матрице:

```
import numpy as np
A = np.arange(9).reshape(3,3)
print(A)
```

```
A[[0,1]] = A[[1,0]]
print(A)
```

Результат: [[0 1 2]
[3 4 5]
[6 7 8]]
[[3 4 5]
[0 1 2]
[6 7 8]]

10. Определение ранга матрицы:

```
import numpy as np
Z = np.random.uniform(0,1,(2,2))
print(Z)
rank = np.linalg.matrix_rank(Z)
print(rank)
```

Результат: [[0.59243507 0.98871815]
[0.27355253 0.96881476]]
2

Библиотека SciPy

Библиотека SciPy — один компонентов, который включает средства для обработки числовых последовательностей, лежащих в основе моделей машинного обучения: интеграции, экстраполяции, оптимизации и других. Наиболее часто используется не сама SciPy, а библиотека scikit-learn, которая во многом опирается на неё. SciPy организована в подпакеты, охватывающих различные научные вычислительные области.

Приведём примеры применения данной библиотеки для решения различного года задач:

1. Вычисление интеграла $\int_0^1 ax^2 + bdx$

```
from scipy.integrate import quad
def integrand(x, a, b):
    return a*x**2 + b
a = 2
b = 1
I = quad(integrand, 0, 1, args=(a,b))
print(I)
```

Результат: (1.6666666666666667, 1.8503717077085944e-14)

2. Вычисление детерминанта матрицы:

```
import numpy as np
from scipy import linalg
arr = np.array([[1, 2],
                [3, 4]])
print(linalg.det(arr))
```

Результат: -2

3. Вычисление обратной матрицы

```
import numpy as np
from scipy import linalg
arr = np.array([[1, 2],
                [3, 4]])
print(linalg.inv(arr))
```

Результат: [[-2. 1.]
 [1.5 -0.5]]

4. Ресэмплирование сигнала в 25 точек

```
import numpy as np
t = np.linspace(0, 5, 100)
x = np.sin(t)
from scipy import signal
x_resampled = signal.resample(x, 25)
from matplotlib import pyplot as plt
plt.figure(figsize=(5, 4))
plt.plot(t, x, label='Original signal')
plt.plot(t[::4], x_resampled, 'ko',
label='Resampled signal')

plt.legend(loc='best')
plt.show()
```

Результат: см. рис.3.

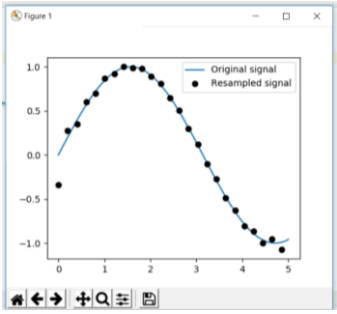


Рис.3. Ресэмплирование сигнала в 25 точек

Matplotlib

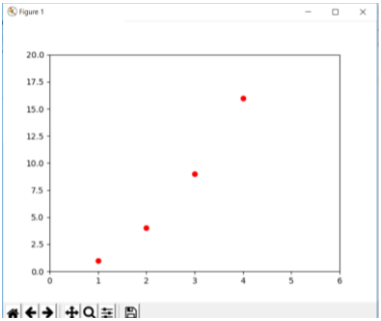
Matplotlib — библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также поддерживается). В данной библиотеке заложены как простые графические команды, так и достаточно сложные. Доступ к ним через означает использование синтаксиса вида "plt.название_команды()".

Приведём примеры использования данной библиотеки:

1. Нанесение точек на координатную плоскость:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

Результат: см. рис.4



(Рис.4. Точки на координатной плоскости)

2. Круговая диаграмма:

```
import matplotlib.pyplot as plt
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
colors = ['yellowgreen', 'gold',
'lightskyblue', 'lightcoral']
explode = (0, 0.1, 0, 0)
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
        autopct='%1.1f%%', shadow=True,
startangle=90)
plt.axis('equal')
plt.show()
```

Результат: см. рис.5

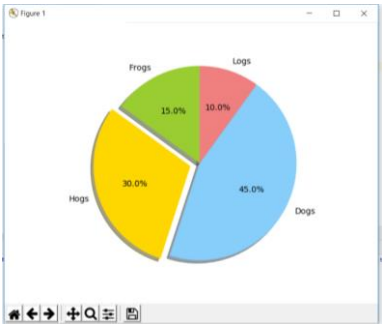


Рис.5. Отображение круговой диаграммы

3. Столбчатая диаграмма:

```
import matplotlib.pyplot as plt
import numpy as np

s = ['one','two','three ','four' ,'five']
x = [1, 2, 3, 4, 5]
z = np.random.random(100)
z1 = [10, 17, 24, 16, 22]
z2 = [12, 14, 21, 13, 17]

# bar()
fig = plt.figure()
```

```
plt.bar(x, z1)
plt.title('Simple bar chart')
plt.grid(True)    # линии вспомогательной сетки
plt.show()
```

Результат: см. рис.6.

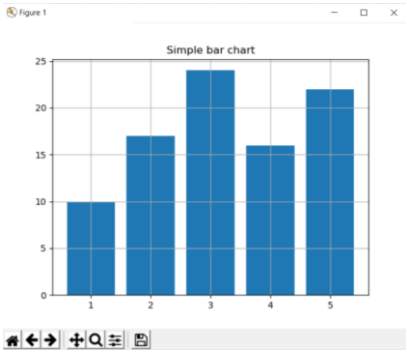


Рис.6. Столбчатая диаграмма

4. Гистограмма:

```
import matplotlib.pyplot as plt
import numpy as np
s = ['one','two','three ','four' ,'five']
x = [1, 2, 3, 4, 5]
z = np.random.random(100)
z1 = [10, 17, 24, 16, 22]
z2 = [12, 14, 21, 13, 17]
fig = plt.figure()
plt.hist(z)
plt.title('Simple histogramm')
plt.grid(True)
plt.show()
```

Результат: см.Рис.7.

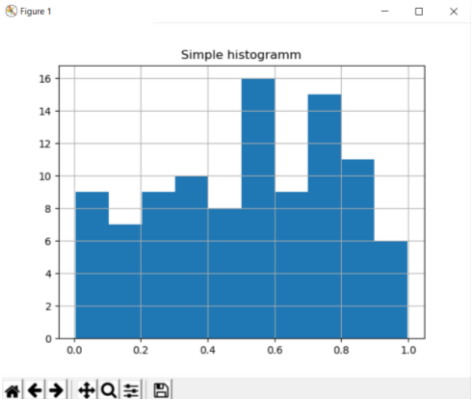


Рис.7. Гистограмма

5. Псевдоцветное отображение значений матрицы:

```
import matplotlib.pyplot as plt
import numpy as np

dat = np.random.random(200).reshape(20,10)  # создаём матрицу значений
cf = plt.matshow(dat)
plt.colorbar(cf, shrink=0.7)
plt.title('Simple matshow plot')
plt.show()
```

Результат: см. рис.8

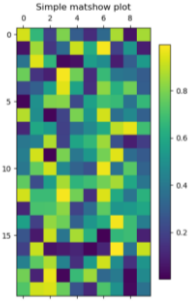


Рис.8. Псевдоцветное отображение матрицы

6. Псевдоцветное отображение синусоиды:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 4*np.pi+0.1, 0.1)
y = np.sin(x)
# fill()
fig = plt.figure()
plt.fill(x, y, 'r') # метод псевдографики
pcolor
plt.title('Simple fill')
plt.grid(True)
plt.show()
```

Результат: см.Рис.9.

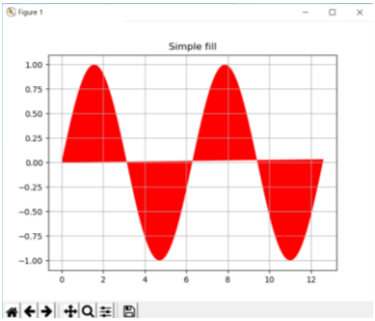


Рис.9. Псевдоцветное Отображение синусоиды

Легенда

Когда на рисунке изображено несколько графиков, возникает потребность в их идентификации. Определить, какая линия или диаграмма соответствует каким данным, помогает легенда.

Легенда - это запись условных обозначениях, располагающаяся на графике. Чтобы повысить читаемость рисунка, нужно стараться делать так, чтобы все использованные условные обозначения легко расшифровывались. Немаловажным аспектом этого является правильное размещение и представление легенды.

Создание легенды подразумевает два логических этапа:

1. Формирование подписей к условным обозначениям;

2. Отображение легенды на рисунке.

Самый простой способ сформировать подписи к графическим элементам - использовать атрибут label в графических командах типа plot(), scatter() и др. Параметр label принимает текст в виде строки, который "привязывается" к породившему их графическому элементу (линия, маркер, диаграмма и т.д.).

Отображение легенды осуществляется либо методом plt.legend() для всего рисунка, либо ax.legend() для конкретной области рисования. Рассмотрим примеры:

```
import matplotlib.pyplot as plt
import pylab
import numpy as np
x = np.arange(0, 4*np.pi+0.1, 0.1)
y = np.sin(x)
# fill()
fig = plt.figure()
plt.fill(x, y, 'r') # метод псевдографики pcolor
plt.title('Simple fill')
plt.grid(True)
pylab.legend ( ("f(x)", ) )
plt.show()
```

Результат: см. рис.10.

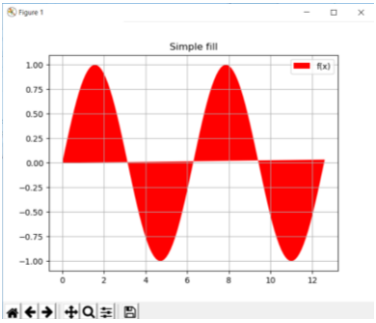


Рис.10. Пример отображения легенды

Pillow

Данная библиотека предназначена для работы с графикой и предоставляет поддержку при открытии, управлении и сохранении многих форматов изображения. Для работы с изображениями удобно поместить их в тот же файл, где расположен проект. Рассмотрим пример использования библиотеки Pillow:

1. Открытие изображения:

```
from PIL import Image
image = Image.open('input.jpg')
image.show()
```

Результат: см. рис.11

Рис.11.Пример открытия изображения

2. Сжатие и сохранение изображения:

```
from PIL import Image
im = Image.open('input.jpg')
im.thumbnail((64, 64))
im.save("output.jpg", "JPEG")
```

Результат: см.Рис.12.

Рис.12.Пример сжатия изображения

3. Размытие изображения:

```
from PIL import Image, ImageFilter
try:
   original = Image.open("input.jpg")
except FileNotFoundError:
   print("Файл не найден")
blurred = original.filter(ImageFilter.BLUR)
original.show()
blurred.show()
blurred.save("blurred.png")
```

Результат: см. рис.13.

Рис.13. Пример сжатия изображения

4. Добавление текста на изображение:

```
from PIL import Image, ImageDraw,ImageFont
NewIm = Image.new('RGB',  (1000,1000),
(128,128,0))
NewImDraw = ImageDraw.Draw(NewIm,'RGB')
image = Image.open("input.jpg")
draw = ImageDraw.Draw(image, "RGBA")
font = ImageFont.truetype("impact.ttf", 70)
draw.text((350, 300), "FLOWER", fill=(40, 0, 0,
15), font=font)
image.save("output.gif")
```

Результат: см. рис.14.

Рис.14. Пример создания надписи

5. Обрезка изображения:

```
from PIL import Image
image = Image.open('input.jpg')
cropped = image.crop((0, 80, 200, 400))
cropped.save('output.png')
```

Результат: см. рис.15.

Рис.15. Пример обрезки изображения

6. Добавление фигур на экран:

```
import random
from PIL import Image, ImageDraw,ImageFont
NewIm = Image.new('RGB',  (1000,1000),
(128,128,0))
NewImDraw = ImageDraw.Draw(NewIm,'RGB')
NewImDraw.ellipse((450,300,950,850),
fill=(255,150,219), outline=None)
NewIm.save('image.png', 'PNG')
```

Результат: см. рис.16.

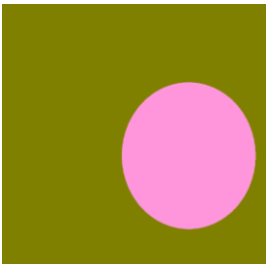


Рис.16.Пример рисования эллипса

7. Обработка: получение изображения в градациях серого, сепия, негатив и добавление шумов:

```
import random
from PIL import Image, ImageDraw
mode = int(input('mode:'))
image = Image.open("000.png")
draw = ImageDraw.Draw(image) #для рисования.
width = image.size[0] #Определяем ширину.
height = image.size[1] #Определяем высоту.
pix = image.load() #Выгрузить значения пикселей
if (mode == 0):
    for i in range(width):
        for j in range(height):
            a = pix[i, j][0]
            b = pix[i, j][1]
            c = pix[i, j][2]
            S = (a + b + c) // 3
            draw.point((i, j), (S, S, S))
if (mode == 1):
    depth = int(input('depth:'))
    for i in range(width):
        for j in range(height):
            a = pix[i, j][0]
            b = pix[i, j][1]
            c = pix[i, j][2]
            S = (a + b + c) // 3
            a = S + depth * 2
            b = S + depth
            c = S
            if (a > 255):
                a = 255
```

```
            if (b > 255):
               b = 255
            if (c > 255):
               c = 255
            draw.point((i, j), (a, b, c))
if (mode == 2):
   for i in range(width):
      for j in range(height):
         a = pix[i, j][0]
         b = pix[i, j][1]
         c = pix[i, j][2]
         draw.point((i, j), (255 - a, 255 - b,
255 - c))
if (mode == 3):
   factor = int(input('factor:'))
   for i in range(width):
      for j in range(height):
         rand = random.randint(-factor, factor)
         a = pix[i, j][0] + rand
         b = pix[i, j][1] + rand
         c = pix[i, j][2] + rand
         if (a < 0):
            a = 0
         if (b < 0):
            b = 0
         if (c < 0):
            c = 0
         if (a > 255):
            a = 255
         if (b > 255):
            b = 255
         if (c > 255):
            c = 255
         draw.point((i, j), (a, b, c))
image.save("rez.jpg", "JPEG")
del draw
```

Результат: см. рис.17

Рис.17. Обработка изображений

Отметим, что описанная выше программа выполняет преобразование изображения в зависимости от введённого при запуске параметра.

Таким образом, можно ввести следующие цифры:

- 0, программа применит к исходному изображению фильтрацию в градациях серого;
- 1, программа применит к исходному изображению фильтрацию “сепия”;
- 2, программа применит к исходному изображению фильтрацию “негатив”;
- 3, программа применит к исходному изображению фильтрацию шум.

ЗАДАЧИ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Рассмотрим выполнение лабораторной работы на примере нескольких задач.

Первая задача: создайте матрицу 4x4, содержащую случайные числа. Умножьте матрицу на цифру 3, затем преобразуйте её элементы и числам типа int. Вычислите ранг и детерминант матрицы, составьте столбчатую диаграмму на основе полученных данных.

Для начала работы необходимо подключить необходимые библиотеки и создать вкладку в новом проекте (в качестве примера используется среда PyCharm). Теперь сделаем импорт необходимых библиотек и модулей.

```
import matplotlib.pyplot as plt
import numpy as np
from scipy import linalg
```

В результате работы программы нам необходимо будет построить столбчатую диаграмму, поэтому создадим для её построения массив из двух элементов. Также создадим необходимую по условию матрицу и умножим её на 3:

```
x = [1, 2]
mas = np.zeros(2)
Z = np.random.random((4,4))
print("Сгенерированная случайным образом матрица")
print(Z)
Z = 3*Z
print("Матрица, умноженная на 3")
print(Z)
```

Приведём значения элементов матрицы к типу int:

```
Z = Z.astype(np.int32, copy=False)
print("Приведение матрицы к целочисленному типу")
print(Z)
```

Вычислим детерминант и ранг полученной в ходе преобразований матрицы и запишем полученные значения в 0 и 1 элемент массива:

```
matr[0] =linalg.det(Z)
print("Определитель первой матрицы")
print(mas[0])
print("Ранг новой матрицы")
matr[1] = np.linalg.matrix_rank(Z)
print(mas[1])
```

Для завершения выполнения задания лабораторной работы создадим столбчатую диаграмму, отражающую полученные значения детерминанта и ранга матрицы:

```
# bar()
fig = plt.figure()
plt.bar(x, matr)
plt.title('Simple bar chart')
plt.grid(True)    # линии вспомогательной сетки
plt.show()
```

Результат: Сгенерированная случайным образом матрица
Сгенерированная случайным образом матрица
[[0.46182182 0.2062318 0.38156297 0.57019351]
 [0.01867229 0.10852546 0.20853909 0.99044133]
 [0.42521417 0.87898401 0.3269458 0.25902999]
 [0.31466105 0.34279622 0.30363946 0.75827485]]
Матрица, умноженная на 3
[[1.38546546 0.61869541 1.14468891 1.71058053]
 [0.05601686 0.32557638 0.62561728 2.97132399]
 [1.2756425 2.63695202 0.9808374 0.77708998]
 [0.94398316 1.02838866 0.91091839 2.27482455]]
Приведение матрицы к целочисленному типу
[[1 0 1 1]
 [0 0 0 2]
 [1 2 0 0]
 [0 1 0 2]]

Определитель первой матрицы
2.0
Ранг новой матрицы
4.0
Столбчатую диаграмму можно увидеть на рис.18

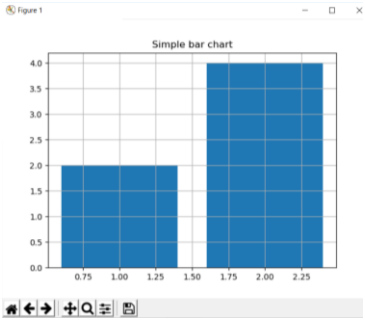


Рис.18. Столбчатая диаграмма

Вторая задача: написать программу, в которой случайным образом будет определяться количество кинотеатров и кафе в городе. На основе этих данных отметить положение каждого из них на карте, используя условные обозначения.

Для начала работы создадим в проекте новую вкладку и подключим необходимые библиотеки:

```
import pylab as pl
import random
import numpy as np
```

Определим случайным образом количество кинотеатров и кафе. В качестве дальнейших условных обозначений будем использовать точки зелёного и синего цветов:

```
r = (random.randint(1, 20))
Z = np.random.random(r)
Z=Z*10
print("Кинотеатр - цвет точки - зелёный - ",+r)
```

```
r1 = (random.randint(1, 10))
Z1 = np.random.random(r1)
Z1=Z1*10
print("Кафе - цвет точки - синий - ",+r1)
```

Выполним отображение положения кинотеатров и кафе на карте:

```
x = range(len(Z))
x1 = range(len(Z1))

pl.plot(x, Z,'go')
pl.plot(x1, Z1,'bo')
pl.show()
```

Результат: Кинотеатр - цвет точки - зелёный - 9
Кафе - цвет точки - синий - 5
Созданную карту можно увидеть на рис.19.

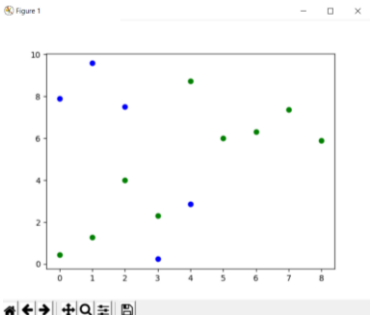


Рис.19.Положение кинотеатров и кафе в городе

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Вариант №1

Задача 1. Создайте матрицу размером 6x6, на границах которой расположены 1, внутри пространство заполнено 0. Создайте вторую матрицу размером 6x6, на границах которой расположены 0, внутри пространство заполнено 1. Перемножьте две созданные матрицы. Для каждой из трёх матриц определите ранг. Составьте круговую диаграмму, отражающую ранги каждой из матриц. От общей диаграммы отделите часть, соответствующую наибольшему значению ранга.

Задача 2. Дано изображение HOUSE.JPG размером 2560x1600 пикселей. Уменьшите изображение в 4 раза. Пересохраните.

Задача 3. С помощью библиотек NumPy и MatplotLib постройте графики двух линейных функций — возрастающей и убывающей, и закрасьте образованные ими углы двумя разными цветами (т.е. вертикальные углы закрашиваются одним цветом).Цвета закрашенных углов не должны совпадать с цветами линий линейных функций.

Вариант № 2.

Задача 1. Создайте матрицу 5x5, содержащую случайные числа. Умножьте матрицу на число 10, затем преобразуйте содержащиеся в матрице значения цифрам и числам типа int. Определите детерминант матрицы. Сохраните полученное значение в зарезервированную переменную. Вычислите обратную матрицу к данной и снова определите её детерминант. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя первой и второй матрицы.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте розовый цветок, подпишите красным цветом FLOWER (шрифт Britannic Bold), сохраните результат на диск, затем откройте и добавьте шум на изображение. Сохраните еще раз

Задача 3. Представьте что Вы-разработчик PostGIS.(Прим.: PostGIS — это географическая информационная система, или просто ГИС (geographic information system, GIS), которая позволяет хранить пространственные или географические данные, такие, как точки, ломаные линии и полигоны, производить по ним эффективный поиск, а также выполнять с ними другие операции). Постройте столбчатую диаграмму, отображающую сколько точек на карте к какому типу (школа, кафе, парк и т.п.) относятся.

Вариант № 3.

Задача 1. Создайте единичную матрицу размером 6x6. Присвойте элементам побочной диагонали значения равные 1. Вычислите ранг матрицы. К нулевым элементам матрицы прибавьте возведённое во 2 степень значение ранга матрицы. Равные единице элементы умножьте на ранг матрицы. С помощью команды для псевдоцветного изображения матрицы отобразите полученную в ходе преобразований матрицу.

Задача 2. Дано изображение CAT.JPG. С помощью инструментов Pillow, сделайте изображение расплывчатым (эффект BLUR), обрежьте его по собственному желанию, сохраните на диск под новым именем CAT_1.JPG

Задача 3. Представьте что Вы-разработчик PostGIS.(Прим.: PostGIS — это географическая информационная система, или просто ГИС (geographic information system, GIS), которая позволяет хранить пространственные или географические данные, такие, как точки, ломаные линии и полигоны, производить по ним эффективный поиск, а также выполнять с ними другие операции). Постройте вертикальную диаграмму, отображающую сколько точек на карте к какому типу (школа, кафе, парк и т.п.) относятся.

Вариант № 4.

Задача 1. Создайте матрицу размером 5x5, на границах которой расположены 0, внутри пространство заполнено 1. Определите значение её ранга. Создайте массив, заполненный числами от 6 до 16. “Разверните” массив. Перемножьте значения седьмого элемента массива и ранга матрицы, запишите в переменную numb. Создайте новую матрицу 4x4, заполненную числами от 0 до numb-1. С помощью команды отображения данных матрицы в виде квадратов отобразите значения созданной матрицы.

Задача 2. Дано изображение BED.JPG. С помощью инструментов Pillow, добавьте эффект негатива и сделайте надпись GOODNIGHT, сохраните на диск под новым именем BED_1.JPG

Задача 3. По умолчанию цвета осей координат, линий вспомогательной сетки отрисовываются чёрным цветом, а цвет фона (основы рисунка) - белым. Измените параметры рисования, отвечающие за соответствующие цвета на ваше усмотрение и выведите на экран график зависимости амплитуды от времени.

Вариант № 5.

Задача 1. Создайте матрицу 5x5, содержащую значения от 0 до 24. Вычислите определитель матрицы, сохраните полученное значение в зарезервированную переменную. Создайте массив со значениями от 2 до 16. Поменяйте знак цифр, которые лежат между 2 и 5. Вычислите среднее значение элементов массива. Вычтите из каждого элемента матрицы полученное среднее значение элементов массива. Вычислите определитель матрицы. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя исходной и итоговой матриц.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте домик с красной крышей, синим фасадом и коричневым забором.

Сделайте надпись черным цветом «HOUSE» (шрифт любой) и поверните ее на 180 градусов. Сохраните изображение на диск.

Задача 3. С помощью библиотеки MatplotLib и круговой диаграммы визуализируйте распределение технических университетов по различным городам России.

Вариант № 6.

Задача 1. Создайте матрицу 5x5, содержащую значения -5 и 7 в шахматном порядке. Вычислите определитель матрицы, сохраните полученное значение в зарезервированную переменную. Создайте массив со значениями от 3 до 14. Поменяйте знак цифр, которые лежат между 6 и 8. Вычислите сумму чётных элементов массива. Вычтите из каждого элемента матрицы значение полученной суммы. Вычислите определитель матрицы. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя исходной и итоговой матриц.

Задача 2. Даны изображения CONCERT.JPG и CAT.JPG. С помощью инструментов Pillow, сделайте из файла CAT.JPG ватермарку и наложите на первое изображение. Сохраните изображение на диск

Задача 3. Найти, используя библиотеку NumPy уравнение параболы (f(x) = ax2 + bx + c), проходящей через точки (1,1) и (-1,1) и касающейся биссектрисы 1-й координатной четверти. Изобразить график параболы и биссектрисы которой она касается. А также 2 точки, через которых проходит парабола.

Вариант № 7.

Задача 1. Создайте матрицу 6x6 со значениями в строках от 0 до 5. С помощью команды для псевдоцветного изображения матрицы отобразите матрицу. Вычислите ранг матрицы. Умножьте значения матрицы на полученное число. Поменяйте местами 1 и 6 строки, 3 и 4 строки. С помощью команды для псевдоцветного изображения матрицы отобразите полученную в ходе преобразований матрицу.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте розовый дом и сделайте изображение серым. Сохраните результат на диск.

Задача 3. С помощью библиотек NumPy и MatplotLib, а также легенды изобразите на графике несколько различных графиков (≥ 3-х), сформируйте подписи к графическим элементам и корректно отобразите все график, чтобы каждый можно было идентифицировать

Вариант № 8.

Задача 1. Создайте матрицу 6х6, содержащую случайные числа. Умножьте матрицу на число 10, затем преобразуйте содержащиеся в матрице значения цифрам и числам типа int. Сложите суммы элементов главной и побочной диагоналей, запишите в зарезервированную переменную. Создайте новую матрицу 5х5, содержащую расположенные в шахматном порядке цифры 8 и 3. Сложите суммы элементов главной и побочной диагоналей, запишите во вторую зарезервированную переменную. Создайте столбчатую диаграмму (bar()), визуализирующую значения первой и второй зарезервированных переменных.

Задача 2. Дано изображение CAR.JPG. С помощью инструментов Pillow, обрежьте изображение так, чтобы была машина, добавьте черно-белый эффект и сделайте надпись GOOD LUCK, сохраните на диск под новым именем CAR_1.JPG. Откройте изображение и уменьшите его по собственному желанию. Пересохраните.

Задача 3. С помощью библиотек NumPy и MatplotLib построить график функции y = x2 на отрезке [-10, 10] с шагом изменения аргумента 0.01. При построении графика использовать параметр, который позволит изменить цвет и тип графика (например не сплошная линия, а точечная)

Вариант №9

Задача 1. Создайте матрицу размером 6x6, на границах которой расположены 1, внутри пространство заполнено 0. Создайте вторую матрицу размером 6x6, на границах которой расположены 0, внутри пространство заполнено 1. Перемножьте две созданные матрицы. Для каждой из трёх матриц определите ранг. Составьте круговую диаграмму, отражающую ранги каждой из матриц. От общей диаграммы отделите часть, соответствующую наибольшему значению ранга.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте розовый цветок, подпишите красным цветом FLOWER (шрифт Britannic Bold), сохраните результат на диск, затем откройте и добавьте шум на изображение. Сохраните еще раз

Задача 3. Представьте что Вы-разработчик PostGIS.(Прим.: PostGIS — это географическая информационная система, или просто ГИС (geographic information system, GIS), которая позволяет хранить пространственные или географические данные, такие, как точки, ломаные линии и полигоны, производить по ним эффективный поиск, а также выполнять с ними другие операции). Постройте вертикальную диаграмму, отображающую сколько точек на карте к какому типу (школа, кафе, парк и т.п.) относятся.

Вариант №10

Задача 1. Создайте матрицу 5x5, содержащую случайные числа. Умножьте матрицу на число 10, затем преобразуйте содержащиеся в матрице значения цифрам и числам типа int. Определите детерминант матрицы. Сохраните полученное значение в зарезервированную переменную. Вычислите обратную матрицу к данной и снова определите её детерминант. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя первой и второй матрицы.

Задача 2. Дано изображение CAT.JPG. С помощью инструментов Pillow, сделайте изображение расплывчатым (эффект BLUR), обрежьте его по собственному желанию, сохраните на диск под новым именем CAT_1.JPG

Задача 3. По умолчанию цвета осей координат, линий вспомогательной сетки отрисовываются чёрным цветом, а цвет фона (основы рисунка) - белым. Измените параметры рисования, отвечающие за соответствующие цвета на ваше усмотрение и выведите на экран график зависимости амплитуды от времени.

Вариант № 11.

Задача 1. Создайте единичную матрицу размером 6х6. Присвойте элементам побочной диагонали значения равные 1. Вычислите ранг матрицы. К нулевым элементам матрицы прибавьте возведённое во 2 степень значение ранга матрицы. Равные единице элементы умножьте на ранг матрицы. С помощью команды для псевдоцветного изображения матрицы отобразите полученную в ходе преобразований матрицу.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте розовый цветок, подпишите красным цветом FLOWER (шрифт Britannic Bold), сохраните результат на диск, затем откройте и добавьте шум на изображение. Сохраните еще раз.

Задача 3. Представьте что Вы-разработчик PostGIS.(Прим.: PostGIS — это географическая информационная система, или просто ГИС (geographic information system, GIS), которая позволяет хранить пространственные или географические данные, такие, как точки, ломаные линии и полигоны, производить по ним эффективный поиск, а также выполнять с ними другие операции). Постройте вертикальную диаграмму, отображающую сколько точек на карте к какому типу (школа, кафе, парк и т.п.) относятся.

Вариант № 12.

Задача 1. Создайте матрицу размером 5x5, на границах которой расположены 0, внутри пространство заполнено 1. Определите значение её ранга. Создайте массив, заполненный числами от 6 до 16. “Разверните” массив. Перемножьте значения седьмого элемента массива и ранга матрицы, запишите в переменную numb. Создайте новую матрицу 4x4, заполненную числами от 0 до numb-1. С помощью команды отображения данных матрицы в виде квадратов отобразите значения созданной матрицы.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте домик с красной крышей, синим фасадом и коричневым забором. Сделайте надпись черным цветом «HOUSE» (шрифт любой) и поверните ее на 180 градусов. Сохраните изображение на диск.

Задача 3. Найти, используя библиотеку NumPy уравнение параболы (f(x) = ax2 + bx + c), проходящей через точки (1,1) и (-1,1) и касающейся биссектрисы 1-й координатной четверти. Изобразить график параболы и биссектрисы которой она касается. А также 2 точки, через которых проходит парабола.

Вариант № 13.

Задача 1. Создайте матрицу 5x5, содержащую значения от 0 до 24. Вычислите определитель матрицы, сохраните полученное значение в зарезервированную переменную. Создайте массив со значениями от 2 до 16. Поменяйте знак цифр, которые лежат между 2 и 5. Вычислите среднее значение элементов массива. Вычтите из каждого элемента матрицы полученное среднее значение элементов массива. Вычислите определитель матрицы. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя исходной и итоговой матриц.

Задача 2. Даны изображения CONCERT.JPG и CAT.JPG. С помощью инструментов Pillow, сделайте из файла CAT.JPG ватермарку и наложите на первое изображение. Сохраните изображение на диск

Задача 3. С помощью библиотек NumPy и MatplotLib, а также легенды изобразите на графике несколько различных графиков (≥ 3-х), сформируйте подписи к графическим элементам и корректно отобразите все график, чтобы каждый можно было идентифицировать

Вариант № 14.

Задача 1. Создайте матрицу 5x5, содержащую значения -5 и 7 в шахматном порядке. Вычислите определитель матрицы, сохраните полученное значение в зарезервированную переменную. Создайте массив со значениями от 3 до 14. Поменяйте знак цифр, которые лежат между 6 и 8. Вычислите сумму чётных элементов массива. Вычтите из каждого элемента матрицы значение полученной суммы. Вычислите определитель матрицы. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя исходной и итоговой матриц.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте розовый дом и сделайте изображение серым. Сохраните результат на диск.

Задача 3. С помощью библиотек NumPy и MatplotLib построить график функции y = x2 на отрезке [-10, 10] с шагом изменения аргумента 0.01. При построении графика использовать параметр, который позволит изменить цвет и тип графика (например не сплошная линия, а точечная)

Вариант № 15.

Задача 1. Создайте матрицу 6x6 со значениями в строках от 0 до 5. С помощью команды для псевдоцветного изображения матрицы отобразите матрицу. Вычислите ранг матрицы. Умножьте значения матрицы на полученное число. Поменяйте местами 1 и 6 строки, 3 и 4

строки. С помощью команды для псевдоцветного изображения матрицы отобразите полученную в ходе преобразований матрицу.

Задача 2. Дано изображение CAR.JPG. С помощью инструментов Pillow, обрежьте изображение так, чтобы была машина, добавьте черно-белый эффект и сделайте надпись GOOD LUCK, сохраните на диск под новым именем CAR_1.JPG. Откройте изображение и уменьшите его по собственному желанию. Пересохраните.

Задача 3. Напишите программу с использованием библиотек NumPy и Matplotlib, которая выводит на экран график сигмоиды и вычислите значение данной функции в различных точках, отобразив их на графике различными цветами.(Прим.: Сигмоида — это гладкая монотонная возрастающая нелинейная функция, имеющая форму буквы «S», которая часто применяется для «сглаживания» значений некоторой величины)

Вариант № 16.

Задача 1. Создайте матрицу 6x6, содержащую случайные числа. Умножьте матрицу на число 10, затем преобразуйте содержащиеся в матрице значения цифрам и числам типа int. Сложите суммы элементов главной и побочной диагоналей, запишите в зарезервированную переменную. Создайте новую матрицу 5x5, содержащую расположенные в шахматном порядке цифры 8 и 3. Сложите суммы элементов главной и побочной диагоналей, запишите во вторую зарезервированную переменную. Создайте столбчатую диаграмму (bar()), визуализирующую значения первой и второй зарезервированных переменных.

Задача 2. Дано изображение HOUSE.JPG размером 2560x1600 пикселей. Уменьшите изображение в 4 раза. Пересохраните

Задача 3. С помощью библиотек NumPy и MatplotLib постройте в полярной системе координат график спирали Архимеда и график,

заданный уравнением полярной розы $p = \sin k * \varphi$, где k = 1 6 ,изменив цвет фона.

Вариант №17

Задача 1. Создайте матрицу размером 6х6, на границах которой расположены 1, внутри пространство заполнено 0. Создайте вторую матрицу размером 6х6, на границах которой расположены 0, внутри пространство заполнено 1. Перемножьте две созданные матрицы. Для каждой из трёх матриц определите ранг. Составьте круговую диаграмму, отражающую ранги каждой из матриц. От общей диаграммы отделите часть, соответствующую наибольшему значению ранга.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте лицо, подпишите шрифтом Bookman Old Style свою фамилию, поверните ее на 270 градусов. После сохранения откройте изображение и примените эффект сепии. Сохраните еще раз.

Задача 3. По умолчанию цвета осей координат, линий вспомогательной сетки отрисовываются чёрным цветом, а цвет фона (основы рисунка) - белым. Измените параметры рисования, отвечающие за соответствующие цвета на ваше усмотрение и выведите на экран график зависимости амплитуды от времени.

Вариант № 18.

Задача 1. Создайте матрицу 5х5, содержащую случайные числа. Умножьте матрицу на число 10, затем преобразуйте содержащиеся в матрице значения цифрам и числам типа int. Определите детерминант матрицы. Сохраните полученное значение в зарезервированную переменную. Вычислите обратную матрицу к данной и снова определите её детерминант. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя первой и второй матрицы.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте домик с красной крышей, синим фасадом и коричневым забором.

Сделайте надпись черным цветом «HOUSE» (шрифт любой) и поверните ее на 180 градусов. Сохраните изображение на диск.

Задача 3. Представьте что Вы-разработчик PostGIS.(Прим.: PostGIS — это географическая информационная система, или просто ГИС (geographic information system, GIS), которая позволяет хранить пространственные или географические данные, такие, как точки, ломаные линии и полигоны, производить по ним эффективный поиск, а также выполнять с ними другие операции). Постройте вертикальную диаграмму, отображающую сколько точек на карте к какому типу (школа, кафе, парк и т.п.) относятся.

Вариант № 19.

Задача 1. Создайте матрицу 5x5, содержащую значения от 0 до 24. Вычислите определитель матрицы, сохраните полученное значение в зарезервированную переменную. Создайте массив со значениями от 2 до 16. Поменяйте знак цифр, которые лежат между 2 и 5. Вычислите среднее значение элементов массива. Вычтите из каждого элемента матрицы полученное среднее значение элементов массива. Вычислите определитель матрицы. Создайте столбчатую диаграмму (bar()), визуализирующую значения определителя исходной и итоговой матриц.

Задача 2. Используя инструменты библиотеки Pillow, нарисуйте розовый дом и сделайте изображение серым. Сохраните результат на диск.

Задача 3. помощью библиотек NumPy и MatplotLib построить график функции y = x2 на отрезке [-10, 10] с шагом изменения аргумента 0.01. При построении графика использовать параметр, который позволит изменить цвет и тип графика (например не сплошная линия, а точечная)

Вариант № 20.

Задача 1. Создайте матрицу 6x6 со значениями в строках от 0 до 5. С помощью команды для псевдоцветного изображения матрицы отобразите матрицу. Вычислите ранг матрицы. Умножьте значения матрицы на полученное число. Поменяйте местами 1 и 6 строки, 3 и 4 строки. С помощью команды для псевдоцветного изображения матрицы отобразите полученную в ходе преобразований матрицу.

Задача 2. Дано изображение HOUSE.JPG размером 2560x1600 пикселей. Уменьшите изображение в 4 раза. Пересохраните.

Задача 3. С помощью библиотек NumPy и MatplotLib постройте графики двух линейных функций — возрастающей и убывающей, и закрасьте образованные ими углы двумя разными цветами (т.е. вертикальные углы закрашиваются одним цветом).Цвета закрашенных углов не должны совпадать с цветами линий линейных функций.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка заданий.
4. UML-диаграммы классов, используемых в программе, для каждого задания.
5. Листинг программы для каждого задания.
6. Результаты выполнения программ.
7. Выводы по работе в целом.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Опишите назначение библиотек в Python, расскажите о преимуществах их использования.
2. Перечислите и раскройте основные типы операций, использующиеся при работе с массивами в NumPy.
3. Запишите фрагмент кода программы на языке Python для получения атрибута: общего размера массива.
4. Раскройте сущность процесса индексации и приведите примеры обращения ко 2 элементу одномерного массива, к последнему элементу одномерного массива, к первому элементу двумерного массива.
5. Перечислите и раскройте основные типы операций, использующиеся в SciPy.
6. Раскройте область использования библиотеки Matplotlib.
7. Опишите преимущества использования легенд при построении графиков.
8. Выделите логические этапы, необходимые при создании легенды.
9. Перечислите и раскройте основные типы операций, использующиеся при работе с изображениями в Pillow.
10. Выполните анализ синтаксической конструкции: import numpy as np.
11. Запишите фрагмент кода программы на языке Python для 1. нанесения трёх точек на координатную плоскость.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Разработать программы согласно варианту.
4. Выполнить тестирование программ.
5. Продемонстрировать работу программ преподавателю.
6. Оформить отчет.
7. Защитить выполненную работу у преподавателя.

ЛАБОРАТОРНАЯ РАБОТА №5
МНОГОПОТОЧНОЕ ПРОГРАММИРОВАНИЕ НА PYTHON

Целью выполнения лабораторной работы является формирование практических навыков многопоточного программирования, разработки и отладки программ, овладение методами и средствами разработки.

Основными задачами выполнения лабораторной работы являются:

1. Изучить особенности создания потоков и процессов;
2. Научиться создавать многопоточные программы;
3. Изучить типовые алгоритмы решения задач с использованием принципов многопоточного программирования.

Результатами работы являются:

1. Реализация разработанных алгоритмов на языке программирования Python;
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Многопоточное программирование. Основные понятия.

До появления многопоточного программирования работа компьютерных программ состояла из последовательности выполнения, а последовательность выполнения выполнялась последовательно в ЦП центрального процессора хоста. Даже если вся программа состоит из нескольких независимых и не связанных между собой подзадач, программа будет выполняться последовательно.

Поскольку параллельная обработка может значительно повысить эффективность всей задачи, введено многопоточное программирование. Что же такое многопоточность?

Многопоточность – свойство платформы или приложения, состоящее в том, то процесс, порожденный в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка. Такие потоки называют также потоками выполнения (от англ. thread of execution); иногда называют «нитями» (букв. пер. англ. thread) или неформально «тредами»

Многопоточные задачи имеют следующие характеристики:

(1) Природа этих задач асинхронна и требует нескольких одновременных транзакций;

(2) Последовательность выполнения каждой транзакции может быть неопределенной, случайной и непредсказуемой.

Такие задачи программирования можно разделить на несколько потоков выполнения, и каждый поток имеет цель, которую необходимо выполнить.

Представим такую ситуацию:

- У вас на руке смарт-часы, которые собирают данные о вашем пульсе, УФ-излучении и движениях. На смарт-часах работает программа, которая обрабатывает эти данные.

- Программа состоит из четырёх функций. Первая собирает данные с датчиков. Три другие обрабатывают эти данные и делают выводы.
- Пока первая функция не собрала нужные данные, ничего другого не происходит.
- Как только данные введены, запускаются три оставшиеся функции. Они не зависят друг от друга, и каждая считает своё.
- Как только все три функции закончат работу, программа выдаёт нужный результат.

А теперь давайте посмотрим, как это выглядит в однопоточной и многопоточной системе(Рис. 1). Видно, что если процессор позволяет делать несколько дел одновременно, в многопоточном режиме программа будет работать быстрее:

Рис. 1. Работа программы в многопоточном режиме

Компьютерные программы — это просто исполняемые объекты в двоичной (или другой) форме, которые находятся на диске. Программы начинают действовать лишь после их загрузки в память и вызова операционной системой.

Процесс — это программа в ходе ее выполнения (в такой форме процессы иногда называют тяжеловесными процессами).

Потоки (иногда называемые легковесными процессами) подобны процессам, за исключением того, что все они выполняются в пределах одного и того же процесса, следовательно, используют один и тот же контекст. Потоки можно рассматривать как своего рода “мини-процессы”, работающие параллельно в рамках основного процесса или основного потока.

Потоки также обычно допускают совместное применение ресурсов, таких как память и данные, в то время как процессы крайне редко делают это. Проще говоря, некий поток является каким- то независимым компонентом вычислений, который аналогичен процессу, однако сам поток внутри некоего процесса может разделять адресное пространство и данные процесса.

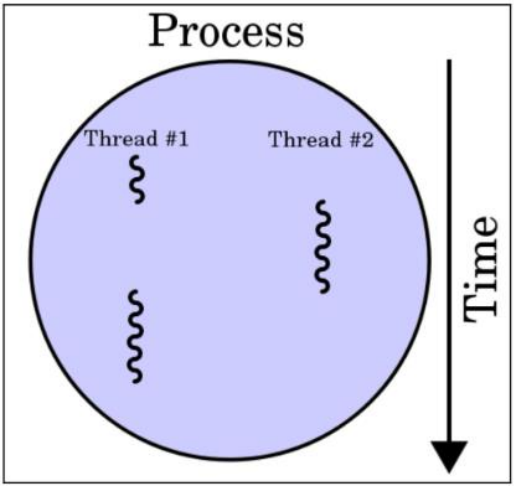


Рис.2. Процесс с двумя потоками исполнения, запущенными на одном процессоре

На одном процессоре многопоточность обычно реализуется путём временного мультиплексирования: процессор переключается между разными потоками выполнения. Это переключение контекста обычно происходит достаточно часто, чтобы пользователь воспринимал выполнение потоков или задач как одновременное.

В многопроцессорных и многоядерных системах потоки или задачи могут реально выполняться одновременно, при этом каждый процессор или ядро обрабатывает отдельный поток или задачу. (Рис. 2.)

Преимущества и недостатки

Многопоточные приложения имеют ряд преимуществ по сравнению с обычными последовательными приложениями; некоторые из них таковы:

- Более быстрое время исполнения: Одним из основных преимуществ совместной обработки при многопоточности является достигаемое ускорение. Отдельные потоки в одной и той же программе могут исполняться совместно или параллельно, когда они достаточно независимы друг от друга.
- Быстрота отклика: Некая программа с единственным потоком за раз может обрабатывать только один кусочек ввода; тем самым, если её основной поток исполнения блокируется в какой- то задаче с длительным временем исполнения (например, некая часть ввода, которая требует интенсивных вычислений и обработки), вся программа целиком не будет способна продолжить прочий ввод и, следовательно, будет казаться замершей. Применяя отдельные потоки для осуществления вычислений и оставаясь исполняемой для получения ввода другого пользователя в то же самое время, некая многопоточная программа может предоставлять гораздо лучшую отзывчивость.
- Эффективность в потреблении ресурсов: как уже упоминалось ранее, множество потоков внутри одного и того же процесса могут совместно разделять одни и те же ресурсы и осуществлять к ним доступ. Вследствие этого, многопоточные программы могут обслуживать и обрабатывать множество запросов клиентов к данным для совместной обработки, используя значительно меньше ресурсов чем это потребовалось бы при применении однопоточных или много процессных программ. Это также ведёт к более быстрому взаимодействию между потоками.

При этом многопоточные программы также имеют и свои собственные недостатки, а именно:

- Крушения: хотя некий процесс может содержать множество потоков, отдельная недопустимая операция в пределах одного потока может отрицательно сказываться на имеющейся обработке всех прочих потоков в этом процессе и может вызывать в результате крушение всей программы целиком.
- Синхронизация: хотя разделение одних и тех же ресурсов может выступать неким преимуществом над обычным последовательным программированием или программами с множеством процессов, для такого совместного использования ресурсов также требуется аккуратное рассмотрение подробностей с тем, чтобы совместные данные вычислялись правильно, и их обработка была корректной. Интуитивно непонятные проблемы, которые могут быть вызваны небрежной координацией потоков включают в свой состав взаимные блокировки, зависания и состояние конкуренции.

Синхронное и асинхронное программирование

Для начала необходимо отметить, что существуют две совершенно разные концепции: первая - синхронная и асинхронная модели программирования, а вторая - однопоточные и многопоточные среды. Каждая из моделей программирования (синхронная и асинхронная) может работать как в однопоточной, так и в многопоточной среде.

Модель синхронного программирования.

В этой модели программирования поток назначается одной задаче и начинает работать над ней. Как только задача завершается, поток доступен для следующей задачи. Т.е. одна задача сменятся другой последовательно. В этой модели невозможно оставить выполнение задачи в середине для выполнения другой задачи. Давайте обсудим, как эта модель работает в однопоточных и многопоточных средах.

Однопоточная среда - Single Threaded - если у нас есть пара задач, которые необходимо выполнить, а текущая система предоставляет

только один поток, тогда задачи назначаются потоку одна за другой. Наглядно это можно изобразить вот так:

Рис.3. Однопоточная среда

Где Thread 1 - один поток, Task 1 и Task 2, Task 3, Task 4 – соответствующие задачи.

Мы видим, что у нас есть поток (Thread 1) и четыре задачи, которые нужно выполнить. Поток начинает работу над задачами и завершает все задачи одну за другой.

Многопоточная среда - Multi-Threaded - в этой среде мы используем несколько потоков, которые могут выполнять эти задачи одновременно. Это означает, что у нас есть пул потоков (новые потоки также могут создаваться по необходимости на основе доступных ресурсов) и множество задач.

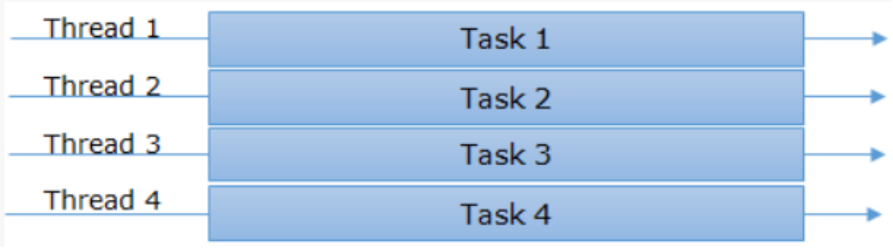


Рис. 4. Многопоточная среда

Мы видим, что у нас есть четыре потока и столько же задач. Поэтому каждый поток выполняет одну задачу и завершает ее. Это идеальный сценарий, но в обычных условиях у нас, как правило, больше задач, чем количество доступных потоков. И поэтому, когда один поток закончит выполнять некоторую задачу, он немедленно приступит к выполнению другой. Обратите внимание также и на тот факт, что новый поток создается не каждый раз, потому что ему нужны системные ресурсы, такие как такты процессора и память, которых может оказаться недостаточно.

Теперь давайте поговорим об асинхронной модели и о том, как она ведет себя в однопоточной и многопоточной среде.

Модель асинхронного программирования.

В отличие от модели синхронного программирования, здесь один поток, запуская некую задачу, может остановить на некотором промежутке времени ее выполнения, сохраняя при этом ее текущее состояние, и начать выполнять другую задачу.

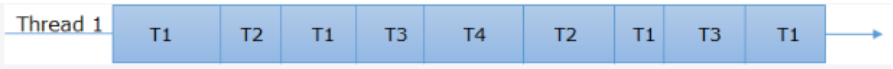


Рис. 5. Модель асинхронного программирования с 1 потоком

мы видим, что один поток отвечает за выполнение всех задач, чередуя их, друг с другом.

Если наша система способна создавать несколько потоков, то все потоки могут работать по асинхронной модели.

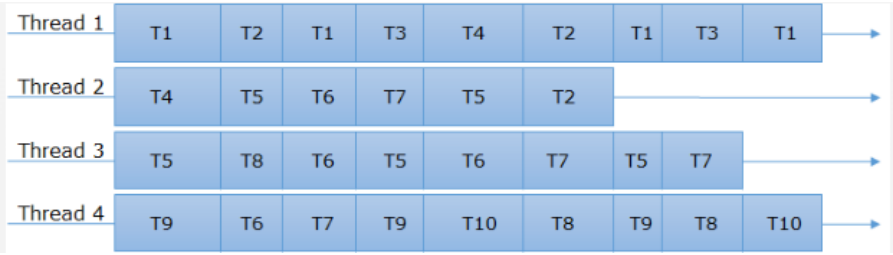


Рис. 6. Модель асинхронного программирования с несколькими потоком

Мы видим, что те же задачи T4, T5, T6 обрабатываются несколькими потоками. В этом и состоит красота и сила этого сценария. Как вы можете видеть, задача T4 была запущена первой в потоке Thread 1 и завершена в потоке Thread 2. Точно так же T6 завершается в Thread 2, Thread 3 и Thread 4.

Итак, всего у нас четыре сценария –

1. Синхронный однопоточный
2. Синхронный многопоточный
3. Асинхронный однопоточный
4. Асинхронный многопоточный

Преимущества асинхронного программирования

Для любого приложения важны две вещи: удобство использования и производительность. Удобство использования важно потому, что

когда пользователь нажимает кнопку, чтобы сохранить некоторые данные, это требует выполнения нескольких небольших задач, таких как чтение и заполнение данных во внутреннем объекте, установление соединения с SQL сервером и сохранение запроса там и. т. д.

Так как SQL-сервер, например, скорее всего, работает на другом компьютере в сети и работает под другим процессом, это может занять много времени. А, если приложение работает в одном потоке, тогда экран устройства пользователя будет находиться в неактивном состоянии до тех пор, пока все задачи не будут завершены, что является примером очень плохого пользовательского интерфейса. Вот почему многие приложения и новые фреймворки полностью полагаются на асинхронную модель, так как она позволяет выполнять множество задач, при этом сохраняя отзывчивость интерфейса.

Эффективность приложения также очень важна. Подсчитано, что при выполнении запроса около 70-80% времени теряется в ожидании зависимых задач. Поэтому, это место, где асинхронное программирование как нельзя лучше придется кстати.

Многопоточное программирование в Python

Python предлагает два модуля для реализации потоков в программах.

- thread
- threading

Для вашей информации, модуль thread устарел в Python 3 и переименован в модуль _thread для обратной совместимости.

Последний модуль threading предоставляет богатые возможности и большую поддержку потоков, чем устаревший модуль thread. Модуль threading является отличным примером многопоточности Python.

Модуль объединяет все методы модуля thread и предоставляет несколько дополнительных методов.

- threading.activeCount(): находит общее число активных объектов потока.

• threading.currentThread(): его можно использовать для определения количества объектов потока в элементе управления потоком вызывающей стороны.

• threading.enumerate(): он предоставит вам полный список объектов потока, которые в данный момент активны.

Помимо описанных выше методов, модуль также представляет класс Thread, который вы можете попробовать реализовать в потоках. Это объектно-ориентированный вариант многопоточности Python.

Класс имеет следующие методы.

Методы класса	Описание метода
run():	Это функция точки входа для любого потока.
start():	запускает поток при вызове метода **run**.
join([time]):	позволяет программе ожидать завершения потоков.
isAlive():	проверяет активный поток.
getName():	извлекает имя потока.
setName():	обновляет имя потока.

При желании вы можете обратиться к родной документации Python, чтобы глубже изучить функциональность модуля threading.

Шаги для реализации потоков с помощью модуля Threading

1. Создайте класс наследовав его от Thread.
2. Переопределите метод __ init __ (self [, args]) для предоставления аргументов в соответствии с требованиями.
3. Затем переопределите метод run(self [, args]), чтобы создать бизнес-логику потока.

Как только вы определили новый подкласс Thread, вы должны создать его экземпляр, чтобы начать новый поток. Затем вызовите

метод для его запуска. В конечном итоге он вызовет метод для выполнения бизнес-логики.

Пример - создание класса потока для печати даты

```
# Пример многопоточности Python для печати текущей даты.
# 1. Определите подкласс, используя класс Thread.
# 2. Создайте подкласс и запустите поток.

import threading
import datetime

class myThread (threading.Thread):
    def __init__(self, name, counter):
        threading.Thread.__init__(self)
        self.threadID = counter
        self.name = name
        self.counter = counter
    def run(self):
        print("Starting " + self.name)
        print_date(self.name, self.counter)
        print("Exiting " + self.name)

def print_date(threadName, counter):
    datefields = []
    today = datetime.date.today()
    datefields.append(today)
    print(
       "%s[%d]:  %s"  %  (  threadName,  counter,
datefields[0] )
    )

# Создать треды
thread1 = myThread("Thread", 1)
thread2 = myThread("Thread", 2)

# Запустить треды
thread1.start()
```

```
thread2.start()

thread1.join()
thread2.join()
print("Exiting the Program!!!")
```

Синхронизация потоков в Python

Для синхронизации доступа к ресурсам из нескольких потоков Python предоставляет набор объектов, каждый из которых обладает рядом особенностей, делающих их пригодными для решения некоторой группы специфических задач. В этом уроке будут рассмотрены: Lock– и RLock-объекты, условные переменные (Condition), семафоры (Semaphore), события (Event), таймеры (Timer) и барьеры (Barrier).

Lock-объект

Lock-объект может находится в двух состояниях: захваченное (заблокированное) и не захваченное (не заблокированное, свободное). После создания он находится в свободном состоянии. Для работы с Lock-объектом используются методы acquire() и release(). Если Lock свободен, то вызов метода acquire() переводит его в заблокированное состояние. Повторный вызов acquire() приведет к блокировке инициировавшего это действие потока до тех пор, пока Lock не будет разблокирован каким-то другим потоком с помощью метода release(). Вывоз метода release() на свободном Lock-объекте приведет к выбросу исключения RuntimeError.

Метод acquire() имеет следующую сигнатуру: acquire(blocking=True, timeout=-1)

Параметры метода:

- blocking

Если параметр равен True, то при вызове на захваченном Lock-объекте выполнение потока остановится, после того как захват будет произведет, метод вернет True. Если параметр равен False, то при вызове на захваченном Lock-объекте поток не будет заблокирован и метод вернет False, если захват будет произведет, то вернет True.

- timeout

Задает время, в течении которого поток будет находиться в заблокированном состоянии при попытке захватить уже занятый Lock-объект. Если в течении заданного времени поток не освободится, то метод вернет значение False.

При успешном захвате Lock-объекта, метод acquire() возвращает значение True.

У Lock-объекта также есть метод locked(), который возвращает True если объект захвачен, False в противном случае.

Освободить Lock-объект может любой поток (не обязательно тот, который вызвал acquire()).

Хорошей практикой при работе с Lock-объектами является помещение кода работы с разделяемым ресурсом в блоке try, а освобождать блокировку следует в finally:

```
lock_obj.acquire()
try:
    # Работа с разделяемым ресурсом
finally:
    lock_obj.release()
```

Lock-объекты поддерживают протокол менеджера контекста, это позволяет работать с ними через оператор with. Приведенный выше код с try…finally эквивалентен следующему:

```
with lock_obj:
    # Работа с разделяемым ресурсом
```

RLock-объект

В отличии от рассмотренного выше Lock-объекта RLock может освободить только тот поток, который его захватил. Повторный захват потоком уже захваченного RLock-объекта не блокирует его. RLock-объекты поддерживают возможность вложенного захвата, при этом освобождение происходит только после того, как был выполнен release() для внешнего acquire(). Сигнатуры и назначение методов release() и acquire() RLock-объектов совпадают с приведенными для

Lock, но в отличии от него у RLock нет метода locked(). RLock-объекты поддерживают протокол менеджера контекста.

Условные переменные (threading.Condition)

Основное назначение условных переменных – это синхронизация работы потоков, которая предполагает ожидание готовности некоторого ресурса и оповещение об этом событии. Наиболее явно такой тип работы выражен в паттерне Producer-Consumer (Производитель – Потребитель). Условные переменные для организации работы внутри себя используют Lock- или RLock-объекты, захватом и освобождением которых управлять не придется, хотя и возможно, если возникнет такая необходимость.

Порядок работы с условными переменными выглядит так:

- На стороне Consumer'а: проверить доступен ли ресурс, если нет, то перейти в режим ожидания с помощью метода wait(), и ожидать оповещение от Producer'а о том, что ресурс готов и с ним можно работать. Метод wait() может быть вызван с таймаутом, по истечении которого поток выйдет из состояния блокировки и продолжит работу.
- На стороне Producer'а: произвести работы по подготовке ресурса, после того, как ресурс готов оповестить об этом ожидающие потоки с помощью методов notify() или notify_all(). Разница между ними в том, что notify() разблокирует только один поток (если он вызван без параметров), а notify_all() все потоки, которые находятся в режиме ожидания.

Ниже представлен пример работы с условной переменной.

```
from threading import Condition, Thread
from queue import Queue
from time import sleep
cv = Condition()
q = Queue()
# Consumer function for order processing
def order_processor(name):
   while True:
```

```
            with cv:
                # Wait while queue is empty
                while q.empty():
                    cv.wait()
                try:
                    # Get data (order) from queue
                    order = q.get_nowait()
                    print(f"{name}: {order}")
                    # If get "stop" message then stop thread
                    if order == "stop":
                        break
                except:
                    pass
                sleep(0.1)
    # Run order processors
    Thread(target=order_processor,          args=("thread
1",)).start()
    Thread(target=order_processor,          args=("thread
2",)).start()
    Thread(target=order_processor,          args=("thread
3",)).start()
    # Put data into queue
    for i in range(10):
        q.put(f"order {i}")
    # Put stop-commands for consumers
    for _ in range(3):
        q.put("stop")
    # Notify all consumers
    with cv:
        cv.notify_all()
```

В этом примере мы создаем функцию order_processor, которая может реализовывать в себе бизнес логику, например, обработку заказа. При этом, если она получает сообщение stop, то прекращает свое выполнение. В главном потоке мы создаем и запускаем три потока для обработки заказов. Запущенные потоки видят, что очередь пуста и “встают на блокировку” при вызове wait(). В главном потоке в очередь добавляются десять заказов и сообщения для остановки обработчиков,

после этого вызывается метод notify_all() для оповещения всех заблокированных потоков о том, что данные для обработки есть в очереди.

При создании объекта Condition вы можете передать в конструктор объект Lock или RLock, с которым хотите работать. Перечислим методы объекта Condition с кратким описанием:

- acquire(*args)

Захват объекта-блокировки.

- release()

Освобождение объекта-блокировки.

- wait(timeout=None)

Блокировка выполнения потока до оповещения о снятии блокировки. Через параметр timeout можно задать время ожидания оповещения о снятии блокировки. Если вызвать wait() на Условной переменной, у которой предварительно не был вызван acquire(), то будет выброшено исключение RuntimeError.

- wait_for(predicate, timeout=None)

Метод позволяет сократить количество кода, которое нужно написать для контроля готовности ресурса и ожидания оповещения. Он заменяет собой следующую конструкцию:

```
while not predicate():
    cv.wait()
```

- notify(n=1)

Снимает блокировку с остановленного методом wait() потока. Если необходимо разблокировать несколько потоков, то для этого следует передать их количество через аргумент n.

- notify_all()

Снимает блокировку со всех остановленных методом wait() потоков.

Семафоры (threading.Semaphore)

Реализация классического семафора, предложенного Дейкстрой. Суть его идеи заключается в том, при каждом вызове метода acquire() происходит уменьшение счетчика семафора на единицу, а при вызове

release() – увеличение. Значение счетчика не может быть меньше нуля, если на момент вызова acquire() его значение равно нулю, то происходит блокировка потока до тех пор, пока не будет вызван release().

Семафоры поддерживают протокол менеджера контекста.

Для работы с семафорами в Python есть класс Semaphore, при создании его объекта можно указать начальное значение счетчика через параметр value. Semaphore предоставляет два метода:

- acquire(blocking=True, timeout=None)

Если значение внутреннего счетчика больше нуля, то счетчик уменьшается на единицу и метод возвращает True. Если значение счетчика равно нулю, то вызвавший данный метод поток блокируется, до тех пор, пока не будет кем-то вызван метод release(). Дополнительно при вызове метода можно указать параметры blocking и timeout, их назначение совпадает с acquire() для Lock.

- release()

Увеличивает значение внутреннего счетчика на единицу.

Существует ещё один класс, реализующий алгоритм семафора BoundedSemaphore, в отличии от Semaphore, он проверяет, чтобы значение внутреннего счетчика было не больше того, что передано при создании объекта через аргумент value, если это происходит, то выбрасывается исключение ValueError.

С помощью семафоров удобно управлять доступом к ресурсу, который имеет ограничение на количество одновременных обращений к нему (например, количество подключений к базе данных и т.п.)

В качестве примера приведем программу, моделирующую продажу билетов: обслуживание одного клиента занимает одну секунду, касс всего три, клиентов пять.

```
from threading import Thread, BoundedSemaphore
from time import sleep, time
ticket_office = BoundedSemaphore(value=3)
def ticket_buyer(number):
    start_service = time()
    with ticket_office:
```

```
            sleep(1)
            print(f"client {number}, service time: {time()
- start_service}")
   buyer = [Thread(target=ticket_buyer, args=(i,)) for i
in range(5)]
   for b in buyer:
      b.start()
```

Вывод программы:

client 0, service time: 1.0011110305786133

client 2, service time: 1.0013604164123535

client 1, service time: 1.001556158065796

client 3, service time: 2.002437114715576

client 4, service time: 2.0027763843536377

Как вы можете видеть, вначале обслуживание получили клиенты с номерами 0, 1, 2 и только после того, как кассы по продаже билетов освободились, были обслужены клиенты 3 и 4.

События (threading.Event)

События по своему назначению и алгоритму работы похожи на рассмотренные ранее условные переменные. Основная задача, которую они решают – это взаимодействие между потоками через механизм оповещения. Объект класса Event управляет внутренним флагом, который сбрасывается с помощью метода clear() и устанавливается методом set(). Потоки, которые используют объект Event для синхронизации блокируются при вызове метода wait(), если флаг сброшен.

Методы класса Event:

- is_set()

Возвращает True если флаг находится в взведенном состоянии.

- set()

Переводит флаг в взведенное состояние.

- clear()

Переводит флаг в сброшенное состояние.

- wait(timeout=None)

Блокирует вызвавший данный метод поток если флаг соответствующего Event-объекта находится в сброшенном состоянии. Время нахождения в состоянии блокировки можно задать через параметр timeout.

Пример работы с Event-объектом:

```
from threading import Thread, Event
from time import sleep, time
event = Event()
def worker(name: str):
   event.wait()
   print(f"Worker: {name}")
# Clear event
event.clear()
# Create and start workers
workers = [Thread(target=worker, args=(f"wrk {i}",))
for i in range(5)]
for w in workers:
   w.start()
print("Main thread")
event.set()
```

Содержимое консоли после вызова приведенной Python-программы:

Main thread
Worker: wrk 1
Worker: wrk 2
Worker: wrk 3
Worker: wrk 4
Worker: wrk 0

Порядок пробуждения потоков при использовании объекта Event никак не регламентируется, поэтому вы можете видеть, что поток с номером 0 закончил работу последним, хотя был запущен первым.

Таймеры (threading.Timer)

Модуль threading предоставляет удобный инструмент для запуска задач по таймеру – класс Timer. При создании таймера указывается функция, которая будет выполнена, когда он сработает. Timer реализован как поток, является наследником от Thread, поэтому для его запуска необходимо вызвать start(), если необходимо остановить работу таймера, то вызовите cancel().

Конструктор класса Timer: Timer(interval, function, args=None, kwargs=None)

Параметры:

- interval

Количество секунд, по истечении которых будет вызвана функция function.

- function

Функция, вызов которой нужно осуществить по таймеру.

- args, kwargs

Аргументы функции function.

Методы класса Timer:

- cancel()

Останавливает выполнение таймера

Пример работы с таймером:

```
from threading import Timer
from time import sleep, time
timer = Timer(interval=3,function=lambda: print("Message from Timer!"))
timer.start()
```

Барьеры (threading.Barrier)

Последний инструмент для синхронизации работы потоков, который мы рассмотрим является Barrier. Он позволяет реализовать алгоритм, когда необходимо дождаться завершения работы группы потоков, прежде чем продолжить выполнение задачи.

Конструктор класса: Barrier(parties, action=None, timeout=None)

Параметры:

- parties

Количество потоков, которые будут работать в рамках барьера.

- action

Определяет функцию, которая будет вызвана, когда потоки будут освобождены (достигнут барьера).

- timeout

Таймаут, который будет использовать как значение по умолчанию для методов wait().

Свойства и методы класса:

- wait(timeout=None)

Блокирует работу потока до тех пор, пока не будет получено уведомление либо не пройдет время указанное в timeout.

- reset()

Переводит Barrier в исходное (пустое) состояние. Потокам, ожидающим уведомления, будет передано исключение BrokenBarrierError.

- abort()

Останавливает работу барьера, переводит его в состояние “разрушен” (broken). Все текущие и последующие вызовы метода wait() будут завершены с ошибкой с выбросом исключения BrokenBarrierError.

- parties

Количество потоков, которое нужно для достижения барьера.

- n_waiting

Количество потоков, которое ожидает срабатывания барьера.

- broken

Значение флага равное True указывает на то, что барьер находится в “разрушенном” состоянии.

Пример работы с классом Barrier:

```
from threading import Barrier, Thread
from time import sleep, time
br = Barrier(3)
```

```
store = []
def f1(x):
   print("Calc part1")
   store.append(x**2)
   sleep(0.5)
   br.wait()
def f2(x):
   print("Calc part2")
   store.append(x*2)
   sleep(1)
   br.wait()
Thread(target=f1, args=(3,)).start()
Thread(target=f2, args=(7,)).start()
br.wait()
print("Result: ", sum(store))
```

Результат работы программы:

Calc part1

Calc part2

Result: 23

Данный объект синхронизации может применяться в случаях когда необходимо дождаться результатов работы всех потоков (как в примере выше), либо для синхронизации процесса инициализации потоков, когда перед стартом их работы требуется, чтобы все потоки выполнили процедуру инициализации.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Вариант 1.

Задание 1.

Поиск всех простых чисел (простым называется число, которое является своим наибольшим делителем) в указанном интервале чисел, разделенном на несколько диапазонов. Обработка каждого диапазона производится в порожденном процессе (потоке).

Задание 2.

Имеются один или несколько производителей, генерирующих данные некоторого типа (записи, символы и т.п.) и помещающих их в буфер, а также единственный потребитель, который извлекает помещенные в буфер элементы по одному. Требуется защитить систему от перекрытия операций с буфером, т.е. обеспечить, чтобы одновременно получить доступ к буферу мог только один процесс (производитель или потребитель). Решить задачу с помощью условных переменных.

Вариант 2.

Задание 1.

Warcraft. Заданное количество юнитов добывают золото равными порциями из одной шахты, задерживаясь в пути на случайное время, до ее истощения. Работа каждого юнита реализуется в порожденном процессе (потоке).

Задание 2.

Синхронизировать три потока одного процесса с помощью оператора lock. Каждый поток формирует 5 чисел для одноименной строки матрицы 3x5. Числа первого потока формируются в диапазоне от 10 до 19, второго потока – от 20 до 29 и третьего потока – от 30 до 39. Процесс управляет всеми потоками и выводит на экран результат их работы.

Вариант 3.

Задание 1.

Винни-Пух и пчелы. Заданное количество пчел добывают мед равными порциями (количество меда ограниченно), задерживаясь в пути на случайное время. Винни-Пух потребляет мед порциями заданной величины за заданное время и столько же времени может прожить без питания. Работа каждой пчелы реализуется в порожденном процессе (потоке).

Задание 2.

Имеется общий ресурс и две группы процессов: читатели(которые могут только читать ресурс, но не изменяют его) и писатели(которые изменяют ресурс). В каждый момент работать с ресурсом может сразу несколько читателей (когда ресурс не изменяется писателями), но не более одного писателя. Необходимо синхронизировать их действия над ресурсом и обеспечить взаимное исключение соответствующих критических секций.

Вариант 4.

Задание 1.

Статистический анализ. Имеется несколько массивов данных (разного размера). Требуется определить математическое ожидание в каждом массиве. Обработка каждого массива выполняется в отдельном процессе (потоке).

Задание 2.

Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы c_{ij} матрицы произведения C = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент c_{ij} матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент c_{ij+1}, если j<k, и c_{i+1k}, если j=k. Выполнив вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Если такой элемент есть, то приступает к его расчету. В противном случае отправляет

(пользовательское) сообщение о завершении своей работы и приостанавливает своё выполнение. Главный поток, получив сообщения о завершении вычислений от всех потоков, выводит результат на экран и запускает поток, записывающий результат в конец файла-протокола. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать возможность поработать всем потокам).Обеспечить синхронизацию потоков.

Вариант 5.

Задание 1.

Контрольная сумма. Для нескольких файлов (разного размера) требуется вычислить контрольную сумму (сумму кодов всех символов файла). Обработка каждого файла выполняется в отдельном процессе (потоке).

Задание 2.

Имеются один или несколько производителей, генерирующих данные некоторого типа (записи, символы и т.п.) и помещающих их в буфер, а также единственный потребитель, который извлекает помещенные в буфер элементы по одному. Требуется защитить систему от перекрытия операций с буфером, т.е. обеспечить, чтобы одновременно получить доступ к буферу мог только один процесс (производитель или потребитель). Решить задачу с помощью условных переменных.

Вариант 6.

Задание 1.

Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота –фиксирована, содержащей произвольное количество единиц (препятствий) в произвольных ячейках.

Стартующие бегуны (процессы, потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер.

Задание 2.

Синхронизировать три потока одного процесса с помощью оператора lock. Каждый поток формирует 5 чисел для одноименной строки матрицы 3x5. Числа первого потока формируются в диапазоне от 10 до 19, второго потока – от 20 до 29 и третьего потока – от 30 до 39. Процесс управляет всеми потоками и выводит на экран результат их работы.

Вариант 7.

Задание 1.

Статистический анализ. Имеется несколько массивов данных (разного размера). Требуется определить среднее значение в каждом массиве, без использования встроенной функции. Обработка каждого массива выполняется в отдельном процессе (потоке).

Задание 2.

Имеется общий ресурс и две группы процессов: читатели(которые могут только читать ресурс, но не изменяют его) и писатели(которые изменяют ресурс). В каждый момент работать с ресурсом может сразу несколько читателей (когда ресурс не изменяется писателями), но не более одного писателя. Необходимо синхронизировать их действия над ресурсом и обеспечить взаимное исключение соответствующих критических секций.

Вариант 8.

Задание 1.

Статистический анализ. Имеется несколько массивов данных (разного размера). Требуется определить максимальное значение в каждом массиве, без использования встроенной функции. Обработка каждого массива выполняется в отдельном процессе (потоке).

Задание 2.

Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы c_{ij}

матрицы произведения C = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент c_{ij} матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент c_{ij+1}, если j<k, и c_{i+1k}, если j=k. Выполнив вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Если такой элемент есть, то приступает к его расчету. В противном случае отправляет (пользовательское) сообщение о завершении своей работы и приостанавливает своё выполнение. Главный поток, получив сообщения о завершении вычислений от всех потоков, выводит результат на экран и запускает поток, записывающий результат в конец файла-протокола. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать возможность поработать всем потокам).Обеспечить синхронизацию потоков.

Вариант 9.

Задание 1.

Статистический анализ. Имеется несколько массивов данных (разного размера). Требуется определить минимальное значение в каждом массиве, без использования встроенной функции. Обработка каждого массива выполняется в отдельном процессе (потоке).

Задание 2.

Рассмотрим парикмахерскую, в которой работает один парикмахер, имеется одно кресло для стрижки и несколько кресел в приемной для посетителей, ожидающих своей очереди. Если в парикмахерской нет посетителей, парикмахер засыпает прямо на своем рабочем месте. Появившийся посетитель должен его разбудить, в результате чего парикмахер приступает к работе. Если в процессе стрижки появляются новые посетители, они должны либо подождать своей очереди, либо покинуть парикмахерскую, если в приемной нет свободного кресла для ожидания. Задача состоит в том, чтобы корректно запрограммировать поведение парикмахера и посетителей

Вариант 10.

Задание 1.

Контрольная сумма. Для нескольких файлов (разного размера) требуется вычислить количество слов в файле. Обработка каждого файла выполняется в отдельном процессе (потоке).

Задание 2.

Имеются один или несколько производителей, генерирующих данные некоторого типа (записи, символы и т.п.) и помещающих их в буфер, а также единственный потребитель, который извлекает помещенные в буфер элементы по одному. Требуется защитить систему от перекрытия операций с буфером, т.е. обеспечить, чтобы одновременно получить доступ к буферу мог только один процесс (производитель или потребитель). Решить задачу с помощью условных переменных.

Вариант 11.

Задание 1.

Контрольная сумма. Для нескольких файлов (разного размера) требуется вычислить частоту появления заданного символа (символ задается из консоли). Обработка каждого файла выполняется в отдельном процессе (потоке).

Задание 2.

Синхронизировать три потока одного процесса с помощью оператора lock. Каждый поток формирует 5 чисел для одноименной строки матрицы 3x5. Числа первого потока формируются в диапазоне от 10 до 19, второго потока – от 20 до 29 и третьего потока – от 30 до 39. Процесс управляет всеми потоками и выводит на экран результат их работы.

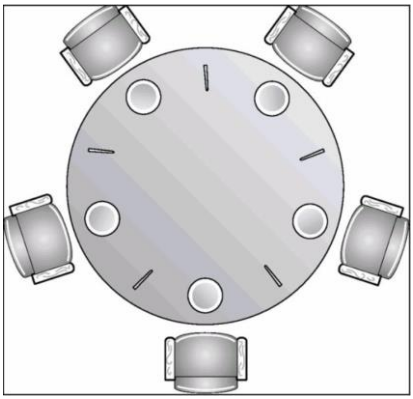
Вариант 12.

Задание 1.

Дано 5 чисел, для каждого числа в отдельном потоке посчитать факториал.

Задание 2.

Имеется круглый стол, за которым сидят пять философов (впрочем, их число принципиального значения не имеет – для другого числа философов решение будет аналогичным). Перед каждым из них лежит тарелка с едой, слева и справа от каждого – две китайские палочки. Философ может находиться в трех состояниях: проголодаться, думать и обедать. На голодный желудок философ думать не может. Но начать обедать он может, только если обе палочки слева и справа от него свободны. Требуется синхронизировать действия философов. В данном случае общим ресурсом являются палочки. Иллюстрацией условий задачи является рисунок.



Вариант 13.

Задание 1.

Дано 5 чисел, для каждого числа n в отдельном потоке определить n-е число Фибоначчи

Задание 2.

Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы c_{ij} матрицы произведения C = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент c_{ij} матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент c_{ij+1}, если j<k, и c_{i+1k}, если j=k. Выполнив

вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Если такой элемент есть, то приступает к его расчету. В противном случае отправляет (пользовательское) сообщение о завершении своей работы и приостанавливает своё выполнение. Главный поток, получив сообщения о завершении вычислений от всех потоков, выводит результат на экран и запускает поток, записывающий результат в конец файла-протокола. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать возможность поработать всем потокам). Обеспечить синхронизацию потоков.

Вариант 14.

Задание 1.

Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота –фиксирована, содержащей произвольное количество единиц (препятствий) в произвольных ячейках.

Стартующие бегуны (процессы, потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер.

Задание 2.

Рассмотрим парикмахерскую, в которой работает один парикмахер, имеется одно кресло для стрижки и несколько кресел в приемной для посетителей, ожидающих своей очереди. Если в парикмахерской нет посетителей, парикмахер засыпает прямо на своем рабочем месте. Появившийся посетитель должен его разбудить, в результате чего парикмахер приступает к работе. Если в процессе стрижки появляются новые посетители, они должны либо подождать своей очереди, либо покинуть парикмахерскую, если в приемной нет свободного кресла для ожидания. Задача состоит в том, чтобы корректно запрограммировать поведение парикмахера и посетителей

Вариант 15.

Задание 1.

Винни-Пух и пчелы. Заданное количество пчел добывают мед равными порциями (количество меда ограниченно), задерживаясь в пути на случайное время. Винни-Пух потребляет мед порциями заданной величины за заданное время и столько же времени может прожить без питания. Работа каждой пчелы реализуется в порожденном процессе (потоке).

Задание 2.

Имеются один или несколько производителей, генерирующих данные некоторого типа (записи, символы и т.п.) и помещающих их в буфер, а также единственный потребитель, который извлекает помещенные в буфер элементы по одному. Требуется защитить систему от перекрытия операций с буфером, т.е. обеспечить, чтобы одновременно получить доступ к буферу мог только один процесс (производитель или потребитель). Решить задачу с помощью условных переменных.

Вариант 16.

Задание 1.

Поиск всех простых чисел (простым называется число, которое является своим наибольшим делителем) в указанном интервале чисел, разделенном на несколько диапазонов. Обработка каждого диапазона производится в порожденном процессе (потоке).

Задание 2.

Синхронизировать три потока одного процесса с помощью оператора lock. Каждый поток формирует 5 чисел для одноименной строки матрицы 3x5. Числа первого потока формируются в диапазоне от 10 до 19, второго потока – от 20 до 29 и третьего потока – от 30 до 39. Процесс управляет всеми потоками и выводит на экран результат их работы.

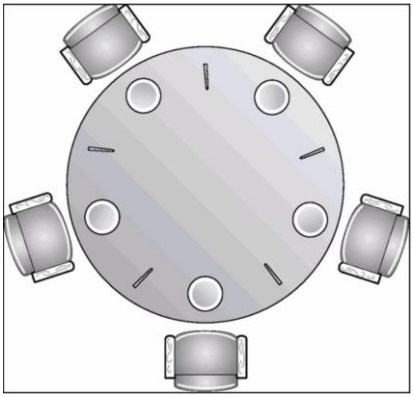
Вариант 17.

Задание 1.

Контрольная сумма. Для нескольких файлов (разного размера) требуется вычислить частоту появления заданного символа (символ задается из консоли). Обработка каждого файла выполняется в отдельном процессе (потоке).

Задание 2.

Имеется круглый стол, за которым сидят пять философов (впрочем, их число принципиального значения не имеет – для другого числа философов решение будет аналогичным). Перед каждым из них лежит тарелка с едой, слева и справа от каждого – две китайские палочки. Философ может находиться в трех состояниях: проголодаться, думать и обедать. На голодный желудок философ думать не может. Но начать обедать он может, только если обе палочки слева и справа от него свободны. Требуется синхронизировать действия философов. В данном случае общим ресурсом являются палочки. Иллюстрацией условий задачи является рисунок.



Вариант 18.

Задание 1.

Статистический анализ. Имеется несколько массивов данных (разного размера). Требуется определить количество четных значений в каждом массиве, без использования встроенной функции.

Обработка каждого массива выполняется в отдельном процессе (потоке).

Задание 2.

Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы c_{ij} матрицы произведения C = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент c_{ij} матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент c_{ij+1}, если j<k, и c_{i+1k}, если j=k. Выполнив вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Если такой элемент есть, то приступает к его расчету. В противном случае отправляет (пользовательское) сообщение о завершении своей работы и приостанавливает своё выполнение. Главный поток, получив сообщения о завершении вычислений от всех потоков, выводит результат на экран и запускает поток, записывающий результат в конец файла-протокола. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать возможность поработать всем потокам).Обеспечить синхронизацию потоков.

Вариант 19.

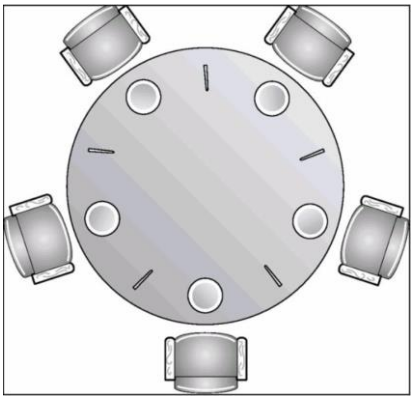
Задание 1.

Статистический анализ. Имеется несколько массивов данных (разного размера). Требуется определить количество не четных значений в каждом массиве, без использования встроенной функции. Обработка каждого массива выполняется в отдельном процессе (потоке).

Задание 2.

Имеется круглый стол, за которым сидят пять философов (впрочем, их число принципиального значения не имеет – для другого числа философов решение будет аналогичным). Перед каждым из них лежит тарелка с едой, слева и справа от каждого – две китайские палочки. Философ может находиться в трех состояниях: проголодаться, думать

и обедать. На голодный желудок философ думать не может. Но начать обедать он может, только если обе палочки слева и справа от него свободны. Требуется синхронизировать действия философов. В данном случае общим ресурсом являются палочки. Иллюстрацией условий задачи является рисунок.



Вариант 20.

Задание 1.

Warcraft. Заданное количество юнитов добывают золото равными порциями из одной шахты, задерживаясь в пути на случайное время, до ее истощения. Работа каждого юнита реализуется в порожденном процессе (потоке).

Задание 2.

Имеется общий ресурс и две группы процессов: читатели(которые могут только читать ресурс, но не изменяют его) и писатели(которые изменяют ресурс). В каждый момент работать с ресурсом может сразу несколько читателей (когда ресурс не изменяется писателями), но не более одного писателя. Необходимо синхронизировать их действия над ресурсом и обеспечить взаимное исключение соответствующих критических секций.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка заданий.
4. Временная UML-диаграмма для созданных потоков в программе для каждого задания.
5. Листинг программы для каждого задания.
6. Результаты выполнения программ.
7. Выводы по работе в целом.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение термину многопоточность.
2. Дайте определение термину поток.
3. Дайте определение термину процесс.
4. Приведите преимущества и недостатки многопоточных программ.
5. Расскажите о моделях синхронного программирования.
6. Расскажите о моделях асинхронного программирования.
7. Опишите методы класса Thread.
8. Расскажите про применение синхронизации потоков.
9. Расскажите о применении и особенностях Lock-объект.
10. Расскажите о применении и особенностях RLock-объекта.
11. Расскажите о применении и особенностях условных переменных.
12. Расскажите о применении и особенностях семафора.
13. Расскажите о применении и особенностях события.
14. Расскажите о применении и особенностях таймера.
15. Расскажите о применении и особенностях Barrier.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 10 академических часа: 9 часов на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Разработать программы согласно варианту.
4. Выполнить тестирование программ.
5. Продемонстрировать работу программ преподавателю.
6. Оформить отчет.
7. Защитить выполненную работу у преподавателя.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Буйначев, С.К. Основы программирования на языке Python: учебное пособие / С.К. Буйначев, Н.Ю. Боклаг; Министерство образования и науки Российской Федерации, Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. - Екатеринбург: Издательство Уральского университета, 2014. - 92 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=275962

2. Саммерфилд, М. Python на практике [Электронный ресурс] : учебное пособие / М. Саммерфилд ; пер. с англ. Слинкин А.А.. — Электрон. дан. — Москва: ДМК Пресс, 2014. — 338 с. — Режим доступа: https://e.lanbook.com/book/66480

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

3. Сузи, Р.А. Язык программирования Python : курс / Р.А. Сузи. - 2-е изд., испр. - Москва : Интернет-Университет Информационных Технологий, 2007. - 327 с. - (Основы информационных технологий). – Режим доступа: http://biblioclub.ru/index.php?page=book&id=233288

4. Хахаев, И.А. Практикум по алгоритмизации и программированию на Python : курс / И.А. Хахаев. - 2-е изд., исправ. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 179 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=429256

Электронные ресурсы:

5. Электронно-библиотечная система http://biblioclub.ru/
6. Электронно-библиотечная система http://e.lanbook.com
7. Сайт о программировании https://metanit.com/