

Колледж космического машиностроения и технологий

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей

программного обеспечения для компьютерных систем

специальность 09.02.03 Программирование в компьютерных системах

Выполнил студент:

Константинович А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2020

Содержание отчёта.

[Раздел 1. Техника решения задач с использованием структурного программирования. 3](#_Toc58928337)

[1.1. Установка интерпретатора Python 3 и настройка окружения 3](#_Toc58928338)

[1.2. Техника работы в командной строке и среде IDLE 9](#_Toc58928339)

[1.3. Техника работы с линейными и разветвляющимися программами. 10](#_Toc58928340)

[1.4. Техника работы с циклическими программами, цикл while. 13](#_Toc58928341)

[1.5. Техника работы с числами. 17](#_Toc58928342)

[1.6. Техника работы со строками. 18](#_Toc58928343)

[1.7. Техника работы со списками. 19](#_Toc58928344)

# **Раздел 1. Техника решения задач с использованием структурного программирования.**

## Установка интерпретатора Python 3 и настройка окружения

Для установки интерпретатора Python на компьютер, первое, что нужно сделать – это скачать дистрибутив. Загрузить его  можно с официального сайта, перейдя по ссылке <https://www.python.org/downloads/>



Рисунок 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.

2. Выбрать способ установки.



Рисунок 2. Установщик Python

3. Отметить необходимые опции установки (доступно при выборе Customize installation)



Рисунок 3. Опции установки

На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Выбираю:

* Documentation – установка документаций.
* pip – установка пакетного менеджера pip.
* tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).

4. Выбираем место установки (доступно при выборе Customize installation)



Рисунок 4. Продвинутые опции установки

5. После успешной установки:



Рисунок 5. Сообщение об установке

Окружение Python представляет собой контекст, в котором выполняется код Python. Различают глобальные, виртуальные окружения и окружения Conda. Окружение состоит из интерпретатора, библиотеки и нескольких установленных пакетов. Вместе они определяют, какие языковые конструкции и синтаксис допустимы, какие возможности операционной системы доступны и какие пакеты можно использовать.

В Visual Studio для Windows есть окно **Окружения Python**, которое позволяет управлять окружениями и выбрать одно из них в качестве окружения по умолчанию для новых проектов.

Окружения, обнаруженные Visual Studio, отображаются в окне **Окружения Python**. Для открытия окна выберите команду меню **Просмотр** > **Другие окна** > **Окружения Python**.



Рисунок 6. Показать Python инструменты



Рисунок 7. Выбор версии Python

При выборе окружения в списке на вкладке **Обзор** Visual Studio отображаются различные свойства и команды для этого окружения. Например, как видно на рисунке выше, интерпретатор находится в папке C:\Python36-32. Четыре команды в нижней части вкладки **Обзор** открывают командную строку с выполняющимся интерпретатором.

Справа от каждого окружения в списке есть элемент управления, который позволяет открыть **интерактивное** окно для этого окружения.

## Техника работы в командной строке и среде IDLE

Выполняя (запуская) команду “python” в вашем терминале, вы получаете интерактивную оболочку Python.



Рисунок 8. Интерактивная оболочка Python

Существует несколько способов закрыть оболочку Python:

>>> exit()

или же

>>> quit()

Кроме того, CTRL + D закроет оболочку и вернет вас в командную строку терминала.

[IDLE](https://docs.python.org/2/library/idle.html) - простой редактор для Python, который поставляется вместе с Python.

Откройте IDLE в вашей системе выбора.

В оболочке есть подсказка из трех прямоугольных скобок:

>>>

Теперь напишите в подсказке следующий код:

>>> print("Hello, World")

Нажмите Enter .

>>> print("Hello, World")

Hello, World



Рисунок 9. Первая программа

## Техника работы с линейными и разветвляющимися программами.

Задание. Разработать программы по темам

- input

- print

- stdin, stdout, stderr модуля sys

- форматная строка и метод формат

Листинг:

import sys

x = int(input())

sys.stdout.write(("{0}, {1}, {2}".format(x-1, x, x+1)))

Задание 1. Разработать программу для печати даты прописью

Пример ввода: 15.12.1983

Пример вывода: Пятнадцатое декабря одна тысяча девятсот восемьдесят третьего года

Листинг:

def getDay(n):

days = ["", "Первое", "Второе", "Третье", "Четвертое", "Пятое", "Шестое", "Седьмое", "Восьмое", "Девятое", "Десятое", "Одиннадцатое",

"Двенадцатое", "Тринадцатое", "Четырнадцатое", "Пятнадцатое", "Шестнадцатое", "Семнадцатое", "Восемнадцатое", "Девятнадцатое", "Двадцатое",

"Двадцать первое", "Двадцать второе", "Двадцать третье", "Двадцать четвертое", "Двадцать пятое"

, "Двадцать шестое", "Двадцать седьмое", "Двадцать восьмое", "Двадцать девятое", "Тридцатое", "Тридцать первое"]

return days[n]

def getMonth(n):

months = ["", "Января", "Февраля", "Марта", "Апреля", "Мая", "Июня", "Июля", "Августа", "Сентября", "Октября", "Ноября",

"Декабря"]

return months[n]

def getYear(n):

result = ""

# get thousands (n <= 10000)

thousandNumbers = ["", "одна тысяча", "две тысячи", "три тысячи", "четыре тысячи", "пять тысяч", "шесть тысяч", "семь тысяч", "восемь тысяч",

"девять тысяч", "десять тысяч"]

lastThousands = ["", "одно тысячного", "двух тысячного", "трех тысячного", "четырех тысячного", "пяти тысячного",

"шести тысячного", "семи тысячного", "восьми тысячного", "девяти тысячного", "десяти тысячного"]

thousands = n // 1000

# get hundreds

hundredsNumbers = ["", "сто", "двести", "триста", "четыреста", "пятьсот", "шестьсот", "семьсот", "восемьсот", "девятсот"]

lastHundreds = ["", "сотого", "двухсотого", "трехсотого", "четырехсотого", "пятисотого", "шестисотого", "семисотого", "восьмисотого", "девятсотого"]

hundreds = n % 1000 // 100

# get dozens and last number

dozensNumbers = ["", "", "двадцать", "тридцать", "сорок", "пятьдесят", "шестьдесят", "семьдесят", "восемьдесят", "девяносто"]

lastDozens = ["" ,"десятого", "двадцатого", "тридцатого", "сорокового", "пятидесятого", "шестидесятого", "семидесятого", "восьмидесятого", "девяностого"]

dozens = n % 100 // 10

oneDozenNumbers = ["", "одиннадцатого", "двенадцатого", "тринадцатого", "четырнадцатого", "пятнадцатого", "шестнадцатого", "семнадцатого",

"восемнадцатого", "девятнадцатого"]

lastNumber = n % 10

lastNumbers = ["", "первого", "второго", "третьего", "четвертого", "пятого", "шестого", "седьмого", "восьмого", "девятого"]

result += "года"

if lastNumber == 0:

result = lastDozens[dozens] + " " + result

elif dozens == 1:

result = oneDozenNumbers[lastNumber] + " " + result

else:

result = lastNumbers[lastNumber] + " " + result

result = dozensNumbers[dozens] + " " + result

if hundreds == 0 and dozens == 0 and lastNumber == 0:

result = lastThousands[thousands] + " " + result

else:

result = hundredsNumbers[hundreds] + " " + result

result = thousandNumbers[thousands] + " " + result

return result

def getDate(d, m, y):

return getDay(d) + " " + getMonth(m) + " " + getYear(y)

day, month, year = map(int, input().split("."))

print(getDate(day, month, year))

Задание 2. Разработать программу с меню для демонстрации работы с типами данных:

список(list), словарь(dict), множество(set)

Меню -> выбор типа данных -> выбор метода -> краткая справка

list\_docs = ["", "Append(x): Add an item to the end of the list.",

"Extend(iterable): Extend the list by appending all the items from the iterable.",

"Insert(i, x): Insert an item at a given position. The first argument is the index of the element before which to insert."]

dict\_docs = ["", "Clear(): Removes all the elements from the dictionary.",

"Values(): Returns a list of all the values in the dictionary.",

"Update({key:value}): Updates the dictionary with the specified key-value pairs."]

set\_docs = ["", "Add(x): Adds an element to the set",

"Discard(x): Remove the specified item",

"Union(...): Return a set containing the union of sets"]

while(1):

print("Выберите структуру данных:")

print("1: List", "2: Dict", "3: Set", "0: Exit", sep='\n')

n = int(input())

if n == 0:

break

print("Выберите метод: ")

if n == 1:

print("1: Append", "2: Extend", "3: Insert", sep='\n')

k = int(input())

print(list\_docs[k])

elif n == 2:

print("1: Clear", "2: Values", "3: Update", sep='\n')

k = int(input())

print(dict\_docs[k])

elif n == 3:

print("1: Add", "2: Discard", "3: Union", sep='\n')

k = int(input())

print(set\_docs[k])

## Техника работы с циклическими программами, цикл while.

Задание 1. На плоскости нарисован квадрат заданного размера с левой нижней

вершиной в начале координат. В квадрат вписывается окружность.

Случайным образом в квадрате выбирается 1000 точек.

а) нужно определить, сколько точек попало внутрь круга

б) считая количество точек пропорциональным площади, найти отношение площадей

круга и квадрата

в) по этому отношению определить приближённое значение числа пи

г) определить, насколько найденное значение отличается от "библиотечного".

Листинг:

import math

from random import randrange

w = int(input())

r = w/2

n = 1000

k = 0

for i in range(n):

x = randrange(0, w)

y = randrange(0, w)

if ((x-r)\*\*2 + (y-r)\*\*2 <= r\*\*2):

k += 1

print("{} точек попало внутрь круга".format(k))

print("Отношение площадей круга и квадрата: {}".format(k/n))

print("Примерное число PI: {}".format(4\*k/n))

print("Разница между полученным и библиотечными числом pi: {}".format(abs(math.pi-4\*k/n)))

Задание 2.

Придумать пример(ы) на использование break / continue /else.

Листинг:

# print first 100 odd numbers

k = 0

x = 0

while True:

if k == 100:

break

if x % 2 == 0:

k += 1

x += 1

print(x)

continue

x += 1

Задание 3. Вычислить значение sin(x) с точностью до epsilon при помощи разложения в ряд. Построить блок-схему.

Листинг:

import math

x = float(input("x: "))

epsilon = float(input("epsilon: "))

sign = 1

\_sum = 0

k = 3

\_next = (x\*\*k)/math.factorial(k)

\_sum = x - \_next

while(\_next > epsilon):

k += 2

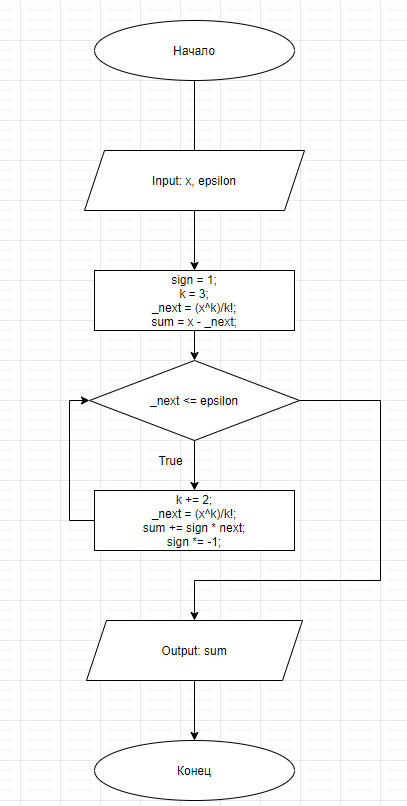
\_next = (x\*\*k)/math.factorial(k)

\_sum += sign \* \_next

sign \*= -1

print(\_sum)

Блок схема:



Задание 4.

https://stepik.org/lesson/3364/step/11?unit=947

Напишите программу, которая считывает со стандартного ввода целые числа, по одному числу

в строке, и после первого введенного нуля выводит сумму полученных на вход чисел.

Sample Input 1:

5

-3

8

4

0

Sample Output 1:

14

Sample Input 2:

0

Sample Output 2:

0

Листинг:

\_sum = 0

n = int(input())

while n != 0:

\_sum += n

n = int(input())

print(\_sum)

Задание 5.

Разработать программу для нахождения наибольшего общего делителя.

Листинг:

def nod (a, b):

return nod (b%a, a) if a else b

a, b = map(int, input().split())

print(nod(a,b))

Задание 6.

С использованием результата задания 2 разработать программу для нахождения наименьшего общего кратного.

Листинг:

def nod(a, b):

return nod(b%a, a) if a else b

def nok(numbers):

while len(numbers) != 1:

a = numbers.pop(0)

b = numbers.pop(0)

c = a\*b/nod(a,b)

numbers.insert(0, c)

return numbers[0]

numbers = []

n = int(input())

while n != 0:

numbers.append(n)

n = int(input())

print(nok(numbers))

Задание 7.

https://stepik.org/lesson/3369/step/8?unit=952

Напишите программу, которая выводит часть последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ...

(число повторяется столько раз, чему равно).

На вход программе передаётся неотрицательное целое число n — столько элементов

последовательности должна отобразить программа.

На выходе ожидается последовательность чисел, записанных через пробел в одну строку.

Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 4.

Sample Input:

7

Sample Output:

1 2 2 3 3 3 4

Листинг:

k = 0

curr = 1

n = int(input())

while k != n:

for i in range(curr):

print(curr, end=' ')

k += 1

if k == n:

break

curr += 1

## Техника работы с числами.

Задание 1. Составить и выполнить по 3 примера использования модулей для работы

с дробными числами (fractions), для точных вычислений (decimal).

Листинг:

####### Fractions library #######

import fractions

# 1 пример: арифметические операции по правилам дробей

a = fractions.Fraction("1/3")

b = fractions.Fraction("2/5")

print(a+b, a-b, a\*b, a/b)

# 2 пример: нахождение НОД

print(fractions.gcd(6,9))

# 3 пример: представление вещественного числа в виде дроби

print(fractions.Fraction("56.45"))

####### Decimal library #######

import decimal

# 1 пример: точные вычисления (0.1 + 0.1 + 0.1 = 0.3)

a = decimal.Decimal("0.1")

print(a + a + a)

# 2 пример: настройка Decimal через getcontext()

print(decimal.Decimal("3")/decimal.Decimal("7"))

decimal.getcontext().prec = 3 # 3 знака после запятой

print(decimal.Decimal("3")/decimal.Decimal("7"))

# 3 пример: представление в виде дроби

print(decimal.Decimal("3.14").as\_integer\_ratio())

## Техника работы со строками.

Задание 1. https://stepik.org/lesson/201702/step/5?unit=175778

С клавиатуры вводятся строки, последовательность заканчивается точкой.

Выведите буквы введенных слов в верхнем регистре, разделяя их пробелами.

Листинг:

s = input()

for c in s:

print(c.upper(), end=' ')

Задание 2. https://stepik.org/lesson/201702/step/8?unit=175778

Известно, что для логина часто не разрешается использовать строки содержащие пробелы.

Но пользователю нашего сервиса особенно понравилась какая-то строка.

Замените пробелы в строке на символы нижнего подчеркивания, чтобы строка

могла сгодиться для логина. Если строка состоит из одного слова, менять ничего не нужно.

Sample Input: python sila

Sample Output: python\_sila

Листинг:

print(input().replace('\_', ' '))

Задание 3. https://stepik.org/lesson/201702/step/9?unit=175778

Уберите точки из введенного IP-адреса. Выведите сначала четыре числа через пробел,

а затем сумму получившихся чисел.

Sample Input: 192.168.0.1

Sample Output:

192 168 0 1

361

Листинг:

bits = tuple(map(int, input().split('.')))

print(\*bits)

print(sum(bits))

Задание 4. https://stepik.org/lesson/201702/step/14?unit=175778

Программист логирует программу, чтобы хорошо знать,

как она себя ведет (эта весьма распространенная и важная практика).

Он использует разные типы сообщений для вывода ошибок (error),

предупреждений (warning), информации (info) или подробного описания (verbose).

Сообщения отличаются по внешнему виду. Назовем модификаторами такие символы,

которые отличают сообщения друг от друга, позволяя программисту понять, к какому

из типов относится сообщения. Модификаторы состоят из двух одинаковых символов

и записываются по разу в начале и в конце строки.

@@ обозначает ошибку

!! обозначает предупреждение

// обозначает информационное сообщение

\*\* обозначает подробное сообщение

Напишите программу, которая принимает строки до точки и выводит,

какого типа это сообщение. Если сообщение не содержит модификаторов,

проигнорируйте его.

Sample Input:

!! cannot resolve this method !!

@@ invalid type @@

@@ StackOverFlowException @@

// here I change the variables name //

\*\* this class is used for operating with the database, including CRUD operations and registering new users \*\*

error on line 42

// TODO: optimize recursive calls //

.

Sample Output:

предупреждение

ошибка

ошибка

информация

подробное сообщение

информация

Листинг:

signals = {"!!" : "предупреждение", "@@" : "ошибка",

"//" : "информация", "\*\*" : "подробное сообщение"}

s = input()

while s != '.':

indicator = s[0:2]

if indicator in signals:

print(signals[indicator])

s = input()

## Техника работы со списками.

Задание 1. https://pythontutor.ru/lessons/lists/problems/more\_than\_neighbours/

Задача «Больше своих соседей»

Дан список чисел. Определите, сколько в этом списке элементов, которые больше двух

своих соседей, и выведите количество таких элементов. Крайние элементы списка никогда

не учитываются, поскольку у них недостаточно соседей.

Листинг:

numbers = list(map(int, input().split()))

k = 0

for i in range(1, len(numbers)-1):

if numbers[i-1] < numbers[i] > numbers[i+1]:

k += 1

print(k)

Задание 2. https://pythontutor.ru/lessons/lists/problems/num\_equal\_pairs/

Задача «Количество совпадающих пар»

Дан список чисел. Посчитайте, сколько в нем пар элементов, равных друг другу.

Считается, что любые два элемента, равные друг другу образуют одну пару, которую

необходимо посчитать.

Листинг:

numbers = list(map(int, input().split()))

k = 0

for i in range(len(numbers)):

for j in range(i+1, len(numbers)):

if numbers[i] == numbers[j]:

k += 1

print(k)

Задание 3. (Л.Б.)

Дано N списков целых чисел (N вводится с клавиатуры, сами списки заполняются

случайным образом). Требуется сформировать

- список, содержащий уникальные значения, попадающие в каждый из N списков

- список, содержащий уникальные значения, попадающие хотя бы в один из N списков

Решение без использования set - дополнительный бонус

Листинг:

import random

allEntry\_array = []

oneEntry\_array = []

n = int(input())

arrays = []

for i in range(n):

arrays.append([random.randint(1, 20) for x in range(30)])

print(arrays[i])

for c in arrays[0]:

inAll = True

if c not in oneEntry\_array:

oneEntry\_array.append(c)

for j in range(1, n):

inCurr = False

for k in arrays[j]:

if k not in oneEntry\_array:

oneEntry\_array.append(k)

if k == c:

inCurr = True

if inAll:

inAll = inCurr

if inAll and c not in allEntry\_array:

allEntry\_array.append(c)

print(allEntry\_array)

print(oneEntry\_array)

Задание 4. Array112. Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки

простым обменом («пузырьковой» сортировкой):

просматривать массив, сравнивая его соседние элементы

(A0 и A1, A1 и A2 и т. д.) и меняя их местами,

если левый элемент пары больше правого; повторить описанные

действия N 1 раз. Для контроля за выполняемыми действиями

выводить содержимое массива после каждого просмотра.

Учесть, что при каждом просмотре количество анализируемых

пар можно уменьшить на 1.

Листинг:

import random

def bubbleSort(A):

l = len(A)

for i in range(l-1):

isSorted = True

for j in range(l-i-1):

if A[j] > A[j+1]:

A[j], A[j+1] = A[j+1], A[j]

isSorted = False

if isSorted:

break

A = [random.randint(1,100) for x in range(20)]

print(\*A)

bubbleSort(A);

print(\*A)

Задание 5. Array113. Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки простым

выбором: найти максимальный элемент массива и поменять его

местами с последним (N-1 м) элементом; выполнить описанные

действия N 1 раз, каждый раз уменьшая на 1 количество

анализируемых элементов и выводя содержимое массива.

Листинг:

import random

def selectSort(A):

l = len(A)

c = l-1

for i in range(l-1):

k = A.index(max(A[:c+1]))

A[k], A[c] = A[c], A[k]

c -= 1

print(\*A)

A = [random.randint(1,100) for x in range(20)]

selectSort(A)

Задание 6. Array114. Дан массив A размера N. Упорядочить

его по возрастанию методом сортировки простыми вставками:

сравнить элементы A0 и A1 и, при необходимости меняя их

местами, добиться того, чтобы они оказались упорядоченными

по возрастанию; затем обратиться к элементу A2 и

переместить его в левую (уже упорядоченную) часть массива,

сохранив ее упорядоченность; повторить этот процесс для

остальных элементов, выводя содержимое массива после

обработки каждого элемента (от 1-го до N-1 го).

Листинг:

import random

# binary find position

def upper\_bound(A, k):

r = len(A)-1

l = 0

while l+1 < r:

m = (l+r) // 2

if A[m] > k:

r = m

else:

l = m

if k < A[l]:

return l

elif k <= A[r]:

return r

else:

return r+1

def insertSort(A):

c = 2

l = len(A)

if A[0] > A[1]:

A[0], A[1] = A[1], A[0]

for i in range(2,l):

A.insert(upper\_bound(A[:c], A[i]), A[i]) #add element to sorted part

A.pop(i+1) #delete this element from previous part

c += 1

print(\*A)

A = [random.randint(1,100) for x in range(20)]

insertSort(A)