Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №1

По теме “Определение модели языка. Выбор инструментальной языковой среды.”

Выполнил:

студент гр. 753504

Осипик И. Ф.

Проверил:

Ст. преподаватель КИ Шиманский В. В.

Минск 2020

Содержание

[**1. Цель работы**](#_k5vdaq8hmnxz) **3**

[**2. Подмножество языка программирования**](#_1fob9te) **4**

[**2.1 Числовые и строковые константы**](#_3th6y9ax3o2g) **4**

[**2.2 Типы переменных**](#_57w84elv0kxk) **4**

[**2.3 Условные операторы**](#_i73qw3zbwd8v) **5**

[**3. Инструментальная языковая среда**](#_5swn028am6em) **7**

[**Примечание. Код программ**](#_uyw0m9k4f85u) **8**

# 

# 1. Цель работы

# 

Необходимо определить подмножество языка программирования (типы констант, переменных, операторов и функций). В подмножество как минимум должны быть включены:

- 3-4 типа переменных;

- числовые и текстовые константы;

- условные операторы (**if**...**else,** **case**);

- операторы цикла (**do**...**while**, **for**).

Определение инструментальной языковой среды, т.е. языка программирования и операционной системы для разработки включает:

- язык программирования c указанием версии, на котором ведётся разработка (напр. Python 3.6);

- операционная система (Windows, Linux и т.д.), в которой выполняется разработка;

- компьютер ([PC / Macintosh).](https://otvet.mail.ru/question/43083633)

В данном отчете по лабораторной работе дается полное определение подмножества языка программирования, тексты 2-3-х программ, включающих все элементы этого подмножества. Приводится подробное описание инструментальной языковой среды.

# 

# 2. Подмножество языка программирования

В рамках лабораторных работ будет проведен анализ и построение интерпретатора подмножества языка Python.

Python— это высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Каждое значение в Python имеет тип. Поскольку всё в Python — объекты, типы являются классами, а значения — экземплярами (объектами) этих классов. В подмножестве будут реализованы такие типы:

## 2.1 Числовые и строковые константы

**Числа.** Целые числа, числа с плавающей запятой и комплексные числа относятся к группе чисел. В Python они представлены классами int, float и complex. Целые числа могут быть любой длины, они ограничиваются лишь доступной памятью. Числа с плавающей запятой имеют ограниченную точность. Визуально разницу между целым числом и числом с плавающей запятой можно заметить в консоли по наличию точки: 1 — целое число, 1.0 — с плавающей запятой.

**Строки.** Строка представляет собой последовательность символов. Мы можем использовать одинарные или двойные кавычки для создания строки. Многострочные строки можно обозначить тройными кавычками, ''' или """. Как и в случае со списками и кортежами, мы можем использовать оператор [] и со строками. Стоит отметить, что строки в Python относятся к категории неизменяемых последовательностей, то есть все функции и методы могут лишь создавать новую строку.

## 2.2 Типы переменных

Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения. Поэтому вместо «присваивания значения переменной» лучше говорить о «связывании значения с некоторым именем». В Python имеются встроенные типы: булевый, строка, Unicode-строка, целое число произвольной точности, число с плавающей запятой, комплексное число и некоторые другие. Из коллекций в Python встроены: список, кортеж (*неизменяемый список*), словарь, множество и другие. Все значения являются объектами, в том числе функции, методы, модули, классы.

* Операторы цикла. В интерпретаторе будут реализованы операторы циклов:
* for - выполняет тело цикла, итерируясь по объекту (к примеру, строке или списку);
* while - выполняет тело цикла до тех пор, пока условие цикла истинно;
* break – оператор, который прерывает исполнение цикла;
* continue - начинает следующий проход цикла, не исполняя оставшееся тело цикла.

## Условные операторы.

В интерпретаторе будут реализованы условные операторы if – elif – else:

* Оператор *if*

if выражение:  
 инструкция\_**1**  
 ...  
 инструкция\_n

* Конструкция if - else

if выражение:  
 инструкция\_1  
 ...  
 инструкция\_n  
else:  
 инструкция\_y  
 ...  
 инструкция\_x

# 3. Инструментальная языковая среда

Для разработки интерпретатора подмножества языка Python будет использован язык программирования C# Разработка проекта будет вестись в среде Visual Studio.

Операционной системой будет выступать Windows 10. Платформа – обычный PC. Интерфейс пользователя будет реализован с помощью терминала Windows PowerShell.

В качестве языковой среды выбран язык программирования C#.

Разработка основана на работе с операционной системой Windows на PC.

Язык **C#** — это объектно-ориентированный язык программирования для разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к С++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

# 

# Примечание. Код программ

3.2.1. Сортировка слиянием

alist = list(54, 26, 93, 17)

def mergeSort(alist) :

print("Splitting ",alist)

if len(alist > 1) :

mid = len(alist) / 2

lefthalf = alist( :mid )

righthalf = alist( mid: )

mergeSort(lefthalf)

mergeSort(righthalf)

i = 0 + 9

j = 0

k = 0

while i < len(lefthalf) and j < len(righthalf) :

if lefthalf(i) < righthalf(j) :

alist(k) = lefthalf(i)

i = i + 1

else :

alist(k+ \*m) = righthalf(j)

j = j + 1

k = k + 1

while i < len(lefthalf) :

alist(k) = lefthalf(i)

i = i + 1

k = k + 1

while j < len(righthalf) :

alist(k) = righthalf(j)

j = j + 1

k = k + 1

print("Merging ",alist)

mergeSort(alist)

print(alist)

3.2.2. Программа работы со списками

>>> a = [66.25, 333, 333, 1, 1234.5]

>>> print(a.count(333), a.count(66.25), a.count('x'))

2 1 0

>>> a.insert(2, -1)

>>> a.append(333)

>>> a

[66.25, 333, -1, 333, 1, 1234.5, 333]

>>> a.index(333)

1

>>> a.remove(333)

>>> a

[66.25, -1, 333, 1, 1234.5, 333]

>>> a.reverse()

>>> a

[333, 1234.5, 1, 333, -1, 66.25]

>>> a.sort()

>>> a

[-1, 1, 66.25, 333, 333, 1234.5]