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. 853505

Лазарева Е. В.

Проверил:

Ст. преподаватель КИ Шиманский В. В.

Минск 2021

Содержание

1. **Цель работы**
2. **Подмножество языка программирования**
   1. **Числовые и строковые константы**
   2. **Типы переменных**
   3. **Условные операторы**
3. **Инструментальная языковая среда**

**Примечание. Код программы**

1. Цель работы

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

- числовые и текстовые константы;

- 3-4 типа переменных;

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

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

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 3.7);

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

- компьютер (PC / Macintosh).

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.

1. Подмножество языка программирования

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

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нем программ. Язык является полностью объектно-ориентированным – все является объектами. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объем полезных функций.

Необычной особенностью языка является выделение блоков кода пробельными отступами. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нем программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как Си или C++.

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

* 1. Числовые и строковые константы

**Числа.** Числа в языке Python представлены тремя встроенными типами: целые (int), вещественные (float) и комплексные (complex). Целые числа могут быть любой длины и ограничиваются только доступной памятью. Числа с плавающей точкой имеют ограниченную точность. Визуально разницу между целым числом и числом с плавающей точкой можно заметить в консоли по наличию точки: 1 – целое число, 1.0 или 1. – число с плавающей точкой.

**Строки.** Строка представляет собой последовательность символов. Для создания строки можно использовать одинарные или двойные кавычки, а многострочные строки можно обозначить тройными кавычками ''' или """. Также возможно использование оператора [] со строками. Стоит отметить, что строки в Python относятся к категории неизменяемых последовательностей, то есть все функции и методы могут лишь создавать новую строку.

* 1. 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». К примитивным типам в Python относятся булевый, целое число произвольной точности, число с плавающей точкой и комплексное число. Из контейнерных типов в Python встроены: строка, список, кортеж (неизменяемый список), словарь и множество. Все значения являются объектами, в том числе функции, методы, модули, классы.

**Операторы цикла.** В интерпретаторе будут реализованы следующие операторы циклов:

* **for** – выполняет тело цикла, итерируясь по объекту (например, списку или строке)
* **while** – выполняет тело цикла до тех пор, пока условие цикла истинно
* **continue** – начинает следующий проход цикла, не исполняя оставшееся тело цикла
* **break** – оператор, который прерывает исполнение цикла
  1. 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\_x

…

инструкция\_y

1. 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.

C# - объектно-ориентированный язык программирования для разработки приложений для платформы Microsoft .NET Framework. 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 C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Delphi, Модула, Smalltalk и, в особенности, Java – C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

Примечание. Код программы

def test1(nums):  
 for i in range(len(nums)):  
 lowest\_value\_index = i  
 for j in range(i + 1, len(nums)):  
 if nums[j] < nums[lowest\_value\_index]:  
 lowest\_value\_index = j  
 nums[i], nums[lowest\_value\_index] = nums[lowest\_value\_index], nums[i]  
 print('Сортировка выборкой')  
 print(nums)  
  
def test2():  
 fib1 = fib2 = 1  
 n = 10  
 print(fib1, fib2, end=' ')  
 for i in range(2, n):  
 fib1, fib2 = fib2, fib1 + fib2  
 print(fib2, end=' ')  
  
def test3(nums):  
 swapped = True  
 while swapped:  
 swapped = False  
 for i in range(len(nums) - 1):  
 if nums[i] > nums[i + 1]:  
 nums[i], nums[i + 1] = nums[i + 1], nums[i]  
 swapped = True  
 print('Сортировка пузырьком')  
 print(nums)  
  
def main\_activity():  
 num = int(input())  
 if num == 1:  
 random\_list\_of\_nums = [12, 8, 3, 20, 11]  
 test1(random\_list\_of\_nums)  
 elif num == 2:  
 print('Нахождение n-го числа Фибоначчи')  
 test2()  
 elif num == 3:  
 random\_list\_of\_nums = [5, 2, 1, 8, 4]  
 test3(random\_list\_of\_nums)  
 else:  
 print('Ошибка! Неверное число!')  
  
main\_activity()

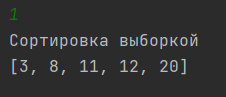


Рис.1 Результат работы функции test1()

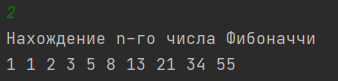


Рис.2 Результат работы функции test2()

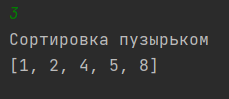


Рис.3 Результат работы функции test3()

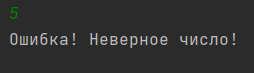


Рис.5 Ошибка при вводе числа больше 3

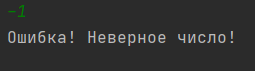


Рис.6 Ошибка при вводе числа меньше 1