Отчет по лабораторной  
  
Вариант 1

* *= 21* – число простых (уникальных) *операторов,* появляющихся в данной реализации;
  + - = 16 – число простых (уникальных) операндов, появляющихся в данной реализации;
    - = 55 – общее число всех операторов, появляющихся в данной реализации;
    - = 36 – общее число всех операндов, появляющихся в данной реализации;
    - – число появлений в программе j-го оператора, где j = 1, 2, 3,…,
    - – число появлений в программе j-го операнда, где j = 1, 2, 3,…, .

Учитывая эти основные метрические характеристики для программы, в конкретной реализации текста программы можно определить:

* + - словарь = 37
    - длину реализации программы = 91
    - длину программы = 156.24
    - объем программы  **= 474**
    - количество имен входных и выходных переменных
    - тогда  **= 6.121**
    - потенциальный объем = 24.54
    - уровень реализации  **= 0.05**

Оптимизация количества и длины модулей в программе

* количество модулей программы k = 1
* словарь операндов одного модуля  **=**
* для всей программы = 15.98
* для всей программы, с учетом того, что каждой группе присваивается имя, которое замещает эту группу в тексте программы  **= 15.998**
* наилучшее число модулей, для обеспечения минимальной длины программы  **1.693**
* тогда число входных модулей **= 3.613**

Оценка уровня языков программирования

* работа программирования  **= 9155.5**
* уровень языка программирования  **= 23.7**
* Существует закон Хика, в соответствии с которым время реакции при выборе из некоего числа альтернативных сигналов зависит от их количества. Тогда по закону Хика работа выбора из библиотеки функций составит , и полная работа программирования при использовании потенциального языка будет определяться следующим образом:  **= 11631.96**
* Квалификационное время программирования **= 508.64**

*Таблица 1.3 Операторы и операции, используемые в программе*

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Операторы,  операции | Номера строк | Количество  повторений |
| 1 | from … | 1, 2 | 2 |
| 2 | = | 4, 5, 6, 7, 9, 13, 19 | 7 |
| 3 | += | 10 | 1 |
| 4 | != | 8 | 1 |
| 5 | // | 6 | 1 |
| 6 | while… | 8 | 2 |
| 7 | ~ | 9 | 2 |
| 8 | return | 15 | 1 |
| 9 | input() | 19 | 1 |
| 10 | print() | 20 | 1 |
| 11 | list() | 4 | 1 |
| 12 | stringIO | 5 | 1 |
| 13 | buffer.write() | 12 | 1 |
| 14 | buffer.getvalue() | 13 | 1 |
| 15 | buffer.close() | 14 | 1 |
| 16 | def | 3, 17 | 3 |
| 17 | for | 11 | 1 |
| 18 | () | 8, 20, 21, 23, 31, 33, 35, 40, 42, 43, 46 | 11 |
| 19 | : | 8, 20, 21, 23, 24, 26, 30, 33, 40, 41, 42, 45 | 12 |
| 20 | if | 45 | 1 |
| 21 | . | 31, 33, 35 | 3 |
| **Всего** | | | 55 |

*Таблица 1.4. Словарь операндов программы*

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Операнды | Номера строк | Количество  повторений |
| 1 | list\_with\_alphas | 4, 9, 11 | 6 |
| 2 | buffer | 5,12 | 2 |
| 3 | StringIO | 5 | 1 |
| 4 | AnyStr | 3, 4 | 3 |
| 5 | string | 3, 4 | 2 |
| 6 | «Введите cтроку для разворота» | 19 | 1 |
| 7 | middle\_index | 6, 8 | 1 |
| 8 | index | 7,8,9,10 | 7 |
| 9 | Reversed\_string | 13, 15 | 1 |
| 10 | reverse | 3, 20 | 2 |
| 11 | s | 19, 20 | 2 |
| 12 | main | 17, 23 | 2 |
| 13 | alpha | 11, 12 | 2 |
| 14 | str | 13, 19 | 2 |
| 15 | name | 35 | 1 |
| 16 | main | 35 | 1 |
|  |  | **Всего** | 36 |

*Таблица 1.6 Значения метрик Холстеда для программы*

|  |  |  |
| --- | --- | --- |
| **Наименование**  **характеристики** | **Обозначение и формула для вычисления** | **Значение** |
| Число простых (уникальных) операторов и операций |  | 21 |
| Число простых (уникальных) операндов |  | 16 |
| Общее число всех операторов и операций |  | 51 |
| Общее число всех операндов |  | 36 |
| Число входных и выходных переменных (параметров) |  | 6.121 |
| Словарь программы |  | 37 |
| Длина реализации программы |  | 91 |
| Объем программы (в битах) |  | 474 |
| Потенциальный объем программы |  | 24.54 |
| Уровень реализации программы |  | 0.05 |
| Уровень реализации языка |  | 1.227 |
| Работа программирования |  | 9155.5 |

**Свой код**

|  |  |
| --- | --- |
| **Номера**  **строк** | **Строки программы** |
| 1 | from io import StringIO |
| 2 | from typing import AnyStr |
| 3 | def reverse(string: AnyStr) -> AnyStr: |
| 4 | list\_with\_alphas: List[AnyStr] = list(string) |
| 5 | buffer: StringIO = StringIO() |
| 6 | middle\_index: int = len(string) // 2 |
| 7 | index: int = 0 |
| 8 | while index != middle\_index: |
| 9 | list\_with\_alphas[index], list\_with\_alphas[~index] = list\_with\_alphas[~index], list\_with\_alphas[index] |
| 10 | index += 1 |
| 11 | for alpha in list\_with\_alphas: |
| 12 | buffer.write(str(alpha)) |
| 13 | reversed\_string: str = buffer.getvalue() |
| 14 | buffer.close() |
| 15 | return reversed\_string |
| 16 |  |
| 17 | def main() -> None: |
| 18 | while True: |
| 19 | s: str = input("Введите строку для разворота") |
| 20 | print(reverse(s)) |
| 21 |  |
| 22 | if \_\_name\_\_ == "\_\_main\_\_": |
| 23 | main() |