Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский

политехнический университет»

(ПНИПУ)

Кафедра вычислительной математики и механики

Отчёт

по дисциплине: «Технологии обработки информации»

по теме: «Код Хаффмена»

Выполнил:

Студент группы ИСТ-19-1бзу

Соболь Евгений Валерьевич

Проверил:

доцент кафедры ВМиМ

Сабатулина Татьяна Леонидовна

Пермь

2021 г.

**Листинг программы с комментариями по каждой функции:**

from collections import Counter  
  
  
class Node(object):  
 *"""Класс узла"""* def \_\_init\_\_(self, name=None, value=None):  
 self.name = name  
 self.value = value  
 self.lchild = None  
 self.rchild = None  
  
  
class HuffmanTree(object):  
 *"""Класс создающий дерево и словарь с значениями кодов"""* def \_\_init\_\_(self, char\_weights):  
 *"""Конструктор класса создающий дерево от корня вверх"""* def out\_leaf(l: list, step: int):  
 *"""Функция вывода созданных узлов"""* if step == 1:  
 print("-----СОЗДАНИЕ УЗЛОВ-----")  
 if step == 2:  
 print("-----ОТСОРТИРУЕМ УЗЛЫ ПО УБЫВАНИЮ-----")  
 if step == 3:  
 print("-----УБЕРАЕМ 2 УБИРАЕМ ДВА УЗЛА С НАИМЕНЬШЕЙ СУММОЙ-----")  
 print(" ДОБАВЛЯЕМ НОВЫЙ УЗЕЛ С ДВУМЯ РОДИТЕЛЯМИ")  
 for \_ in l:  
 print("Узел: ", \_.name)  
 print("Значение: ", \_.value)  
 print("Родитель слева: ", \_.lchild)  
 print("Родитель справа: ", \_.rchild)  
 print("-----------------------")  
 self.Leaf = [Node(k, v) for k, v in char\_weights.items()]  
 out\_leaf(self.Leaf, 1)  
 while len(self.Leaf) != 1:  
 self.Leaf.sort(key=lambda node: node.value, reverse=True)  
 out\_leaf(self.Leaf, 2)  
 n = Node(value=(self.Leaf[-1].value + self.Leaf[-2].value))  
 n.lchild = self.Leaf.pop(-1)  
 n.rchild = self.Leaf.pop(-1)  
 self.Leaf.append(n)  
 out\_leaf(self.Leaf, 3)  
  
 self.root = self.Leaf[0]  
 self.Buffer = list(range(10))  
 self.final\_dict = {}  
  
 def hu\_generate(self, tree, length):  
 *"""Функция рекурсивно проходит по узлам и создает коды символов, сохраняет их в словарь"""* node = tree  
 if not node: # Если нет узла, завершаем рекурсию  
 return  
 elif node.name: # Ecли дошли до основания, записываем код узла  
 kod = ""  
 for i in range(length):  
 kod += str(self.Buffer[i])  
 self.final\_dict[node.name] = kod  
 return  
 self.Buffer[length] = 1  
 self.hu\_generate(node.lchild, length + 1)  
 self.Buffer[length] = 0  
 self.hu\_generate(node.rchild, length + 1)  
  
 def get\_code(self):  
 *"""Вспомогательная функция для генерации словаря с кодами"""* self.hu\_generate(self.root, 0)  
  
  
def output\_dict(d: dict):  
 *"""Функция отображения кодов символов"""* print("--------------------КОДЫ СИМВОЛОВ--------------------")  
 for i in d.keys():  
 print(f'Символ "{i}" - "{d[i]}"')  
 print()  
  
  
def beautiful\_output(d: dict, step: int):  
 *"""Функция для вывода результатов, 1-2, 4 шагов"""* if step == 1:  
 print("-----ПОДСЧЁТ СИМВОЛОВ-----")  
 for i in d:  
 print(f"Символ {i} встречается {d[i]}")  
 print()  
 if step == 2:  
 print("-----ПОДСЧЁТ ЧАСТОТЫ ПОЯВЛЕНИЯ-----")  
 for i in d:  
 print(f"Символ {i} c частотой {d[i]}")  
 print()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print("Задание № 3 Код Хаффмана")  
 source\_line = input("Введите строку: ")  
 final\_dict = dict(Counter(source\_line))  
 beautiful\_output(final\_dict, 1)  
 for i in final\_dict.keys():  
 final\_dict[i] = final\_dict[i] / len(source\_line)  
 beautiful\_output(final\_dict, 2)  
 tree = HuffmanTree(final\_dict)  
 tree.get\_code()  
 output\_dict(tree.final\_dict)  
 result = ""  
 for i in source\_line:  
 result += tree.final\_dict[i]  
 print("--------------------РЕЗУЛЬТАТ--------------------")  
 print(result)

**Пример работы программы № 1:**

Пример 2.4.1 из учебника Кудряшова Б.Д стр. 68

Тестовое значение для ввода: aaaaaaaaaaaaaabbbbbbbbccccccddddeeeeffff

**Результат:**

00000000000000000000000000001010101010101010010010010010010010011011011011110110110110111111111111

**Кодовое дерево кода:**

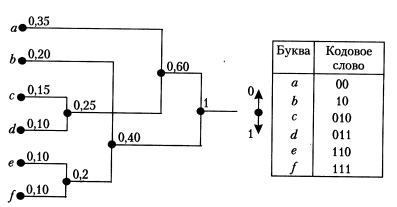
****

Рисунок - Кодовое дерево кода

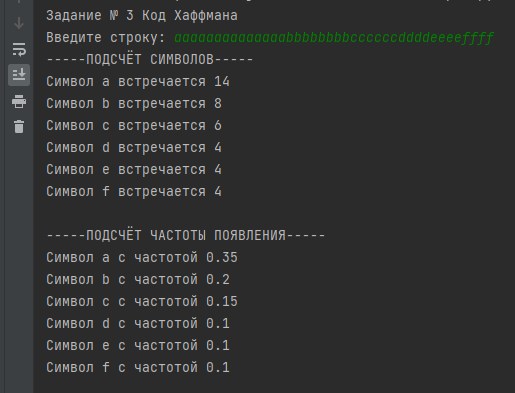


Рисунок – Скриншот программы (1)

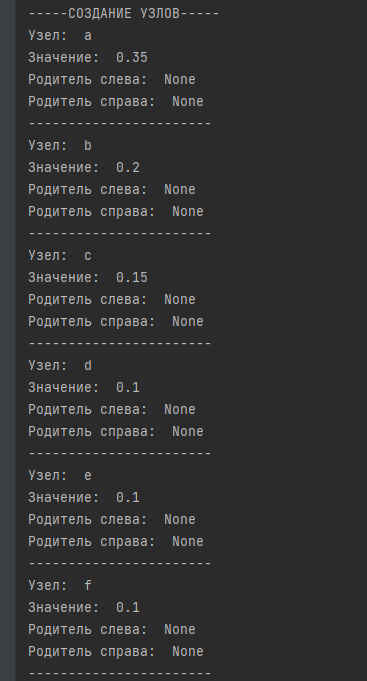


Рисунок – Скриншот программы (2)

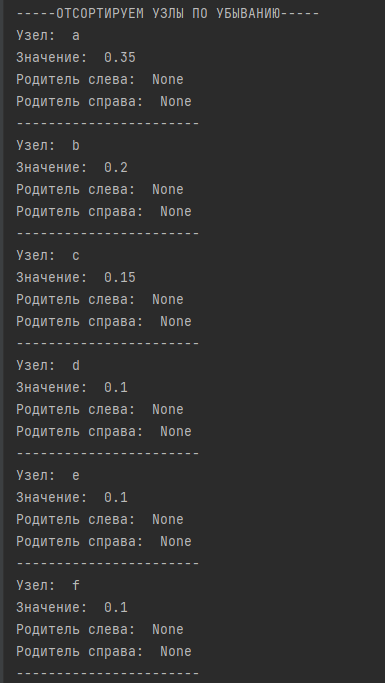


Рисунок – Скриншот программы (3)

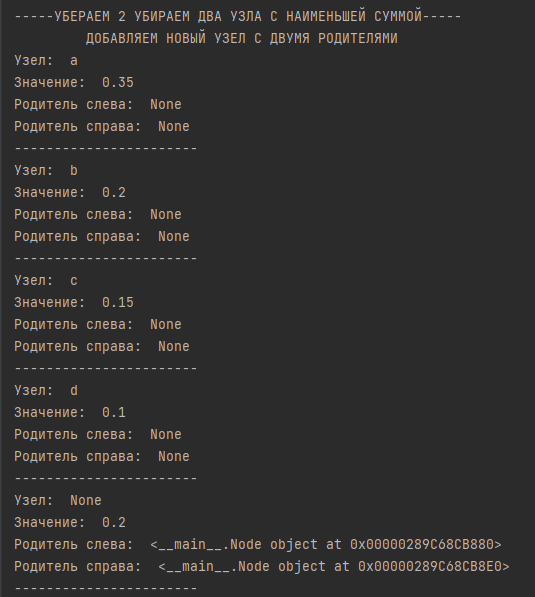


Рисунок – Скриншот программы (4)

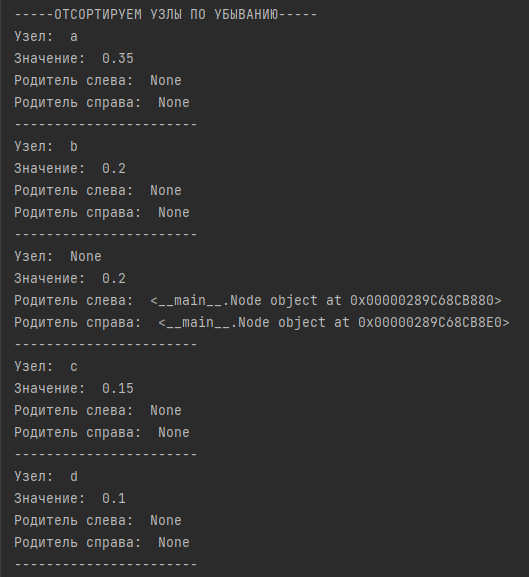


Рисунок – Скриншот программы (5)

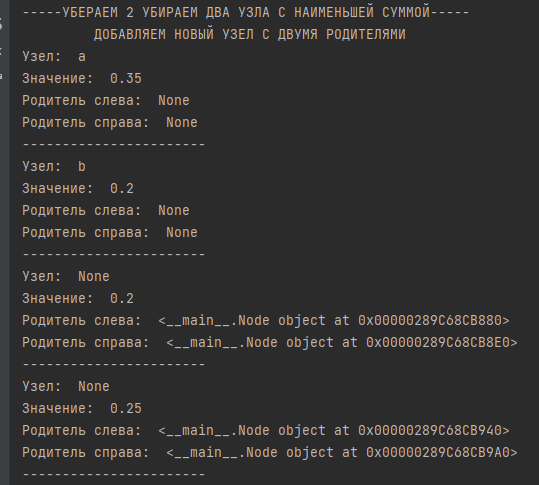


Рисунок – Скриншот программы (6)

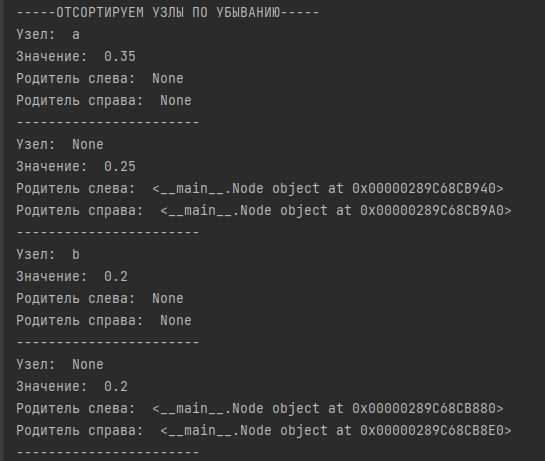


Рисунок – Скриншот программы (7)

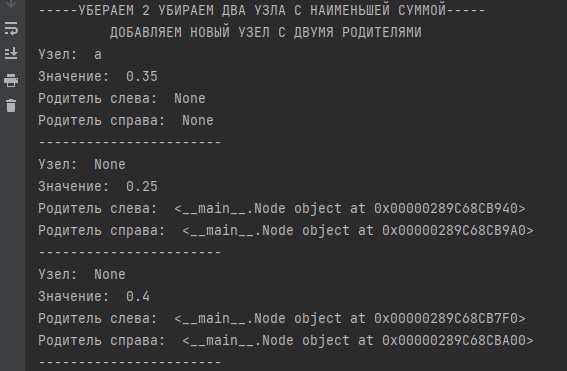


Рисунок – Скриншот программы (8)

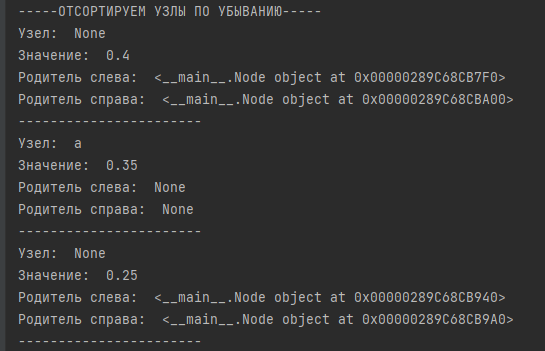


Рисунок – Скриншот программы (9)

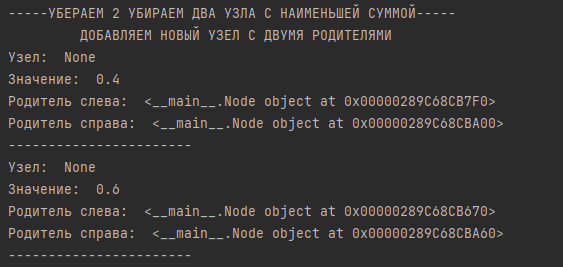


Рисунок – Скриншот программы (10)

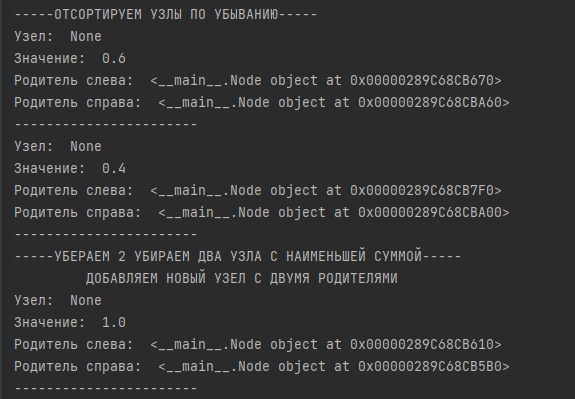


Рисунок – Скриншот программы (11)

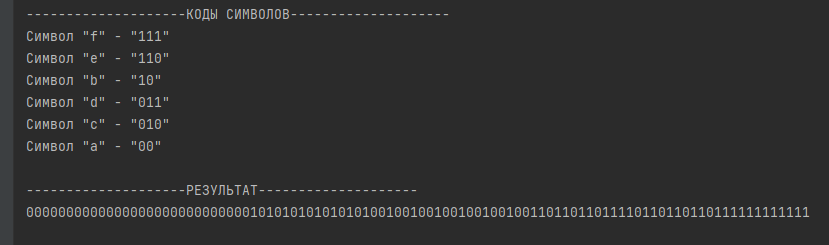


Рисунок – Скриншот программы (12)