Отчет по заданию «Сравнение языков» по дисциплине «Современные языки и технологии программирования»

**Выполнил:** студент 2-го курса Зимин Илья Владимирович

**Учебная группа:** ПМИ – 1

**Вариант:** 1

**Условие задания:** Связный граф задан списком ребер. Каждое ребро представляет собой тройку (вершина, вершина, длина). Граф неориентированный. Найти минимальное остовное дерево (в виде списка образующих его ребер).

**Краткое описание алгоритма:**

Алгоритм был использован один и тот же в обеих реализациях.

Псевдокод:

/\*Инициализация списков с данными:\*/

E //ребра, не включенные в дерево,

uV //вершины, включенные в дерево,

nuV // вершины, не включенные в дерево.

/\*Выбираем произвольную (нулевую вершину) \*/

uV.Add(0) //Добавляем ее в список испольованныъ

nuV.RemoveAt(0) // Убираем из неиспользованных вершин

while nuV.Count не равно 0

min = -1; // Индекс минимального ребра

for (Для каждого ребра из неиспользованных)

if (первая вершина ребра входит в uV а вторая в список nuV

или вторая вершина входит в список uV а вторая в список nuV )

if (min не равно -1)

if (ребро по текущему индексу имеет вес меньше чем ребро E[min])

min = индекс текущего ребра

else min = i

if ( первая вершина E[min] входит в список использованных)

удаляем ее из nuV

добавляем вторую вершину в uV

Иначе наоборот вторую удаляем из nuV а первую добавляем в uV

Добавляем минимальное ребро в граф

Удаляем из Е ребро с индексом min

**Сравнение использованных языков:**

Из объектно-ориентированных языков был выбран язык C#

Из функциональных: Haskell

|  |  |  |
| --- | --- | --- |
| Критерий | C# | Haskell |
| трудоемкость разработки | Не считая разбор алгоритма, на реализацию ушло 1,5 часа:  На подбор структур данных для хранения данных: 30 минут  Набор кода + отладка: 60 минут  Из них набор кода + продумывание реализации методов:  40 минут | Не считая разбор алгоритма, на реализацию ушло 1,3 часа:  На подбор структур данных для хранения данных: 1 минута  Набор кода + отладка: 1 час 17 мин  Из них набор кода + продумывание реализации функций: 50 минут |
| трудоемкость тестирования и отладки | Времени ушло: 20 минут.  Ошибок в ходе тестирования выявлено не было | Времени ушло: 27 минут.  Ошибок в ходе тестирования выявлено не было |
| объем полученного кода | Сам алгоритм:  30 строк.  Вся программа:  54 строки | Сам алгоритм:  9 строк.  Вся программа:  10 строк |
| быстродействие, скорость работы |  |  |

**Небольшие итоги сравнения:**

На мое удивление программу на функциональном языке удалось разработать быстрее, пусть и не на много. Отлаживать было сложнее из-за редактирования кода в блокноте. Кода на языке Haskell вышло в 3 раза меньше, писать его быстрей, но продумывать рекурсивные функции всегда сложнее чем написать цикл, поэтому объектно-ориентированный все же, выиграл в плане скорости написания основной части программы.