1.Описание программных функций

* Создание каталогов локальных дисков с 6-ю каталогами в каждом из них
* Ввод запроса пользователя
* Поиск 5-и каталогов в локальном каталоге по букве локального диска

(Маленький список со следующей информацией: название, год создания и число обращений)

* Сортировка каталога по дате создания и убыванию числа обращений к ним
* Подсчёт среднего количества обращений
* Вывод отсортированного списка каталогов

2.Описание структуры информации

Перечисление типов данных с отсылкой к информации из предметной области

1) Структура Struct Katalog\_info описывающая сведения о каталоге локального диска

string: Название каталога

string: Буква диска

int: Год создания каталога

List<int>: Число обращений к каталогу

2)Множество каталогов

List <Katalog\_info > Katalog

List<Katalog\_info> - все каталоги локального диска:

3)Среднее количество обращений: Double

4) Отфильтрованный список

List<Katalog\_info>

Описание проектирования функций поиска и сортировки

Static List <Katalog\_info> FindByChar(List<Katalog\_info> k, string FilterKat)

Перебор элементов списка k

Получение доступа к конкретному элементу списка

Katalog\_info Katalog = k[i];

3. Обращение к полю буква локального диска в рамках элемента Katalog и сравнение этого значения с запрашивающей буквой локального диска Charrr

If (Katalog. Буква\_локального\_диска== Charrr

Result.Add(Katalog);

Return result; //возврат результата

Функция вычисления среднего количества обращений

double AverageNum (List<int> Число\_обращений\_к\_каталогу)

Складываем все элементы списка в переменную sum и делим на длину списка.

Сортируем маленький список, который нашли с помощью функции find

Определяем функцию, сравнивая каталоги по критерию среднего количества обращений.

int Compare\_Katalog\_By\_Avg Numbers (Katalog\_info k1, Katalog\_info k2)

{

double k1 = AverageNum (k1. Число\_обращений\_к\_каталогу);

double k2 = AverageNum (k2. Число\_обращений\_к\_каталогу);

if k1>k2 return1;

if k1=k2 return 0;

return -1;

}