ОТЧЕТ Лабораторная работа №9 Тема: Шаблонные функции. Группировка элементов массива
Выполнил:Голев Никита Владимирович группа СКБ252

1. Задание Реализовать шаблонную функцию для группировки элементов массива и вычисления статистики по группам. Для каждой группы необходимо определить: key, count, minElem, maxElem, sum, avg.

2. Описание алгоритма
   Программа содержит шаблонную функцию stats, которая принимает массив элементов, его размер и два функтора: keyFunc и valueFunc.
   Функтор keyFunc определяет ключ группы для каждого элемента массива, а valueFunc возвращает значение, используемое для суммирования. Если группа с таким ключом уже существует, её статистика обновляется, иначе создаётся новая группа. После обработки всех элементов вычисляется среднее значение avg для каждой группы.

3. Код программы

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

template <typename S,typename C, typename B> // C – cluch , S – тип элементов масива , B – тип суммы , CS и CB для функторов //
class GroupInfo
{
    public:
        C key;
        int count;
        S minElem;
        S maxElem;
        B sum;
        double avg;
};

// Функторы для всех типов данных //
// Для инта//
class IntKeyFunctor
{
    public:
        int operator()(int x)
        {
            return (x%10);
        }
};
class IntValueFunc
{
    public:
    int operator() (int x) {return x;}
};
//Для дабла//
```

```
class DoubleKeyFunctor
{
    public:
        double operator()(double x)
        {
            if (x<0)
            {
                return 0;
            }
            if (x>10)
            {
                return 2;
            }
            else{
                return 1;
            }
        }
};
class DoubleValueFunc
{
    public:
    double operator() (double x) {return x;}
};

//Для стринга//
class StringKeyFunctor
{
    public:
        char operator()(string x)
        { return x[0];}
};


class StringValueFunc
{
    public:
    int operator() (string x) {return x.size();}
};

// Сама шаблонная функция//
template <typename S, typename C, typename B, typename SC,
typename CB>
vector <GroupInfo<S,C,B>> stats(S arr[],int n, SC keyF, CB valueF)
{
    vector <GroupInfo<S,C,B>> group;
    {
        for (int i=0;i<n;i++)
        {
            S elem = arr[i];
            C curKey = keyF(elem);
            B curVal = valueF(elem);
```

```
            int foundindex = -1;
            for (int j =0 ; j<group.size();j++)
            {
                if (group[j].key == curKey)
                {
                    foundindex = j;
                    break;
                }
            }
            if (foundindex != -1)
            {
                group[foundindex].count++;
                group[foundindex].sum+=curVal;
                if (elem < group[foundindex].minElem)
                {
                    group[foundindex].minElem = elem;
                }
                if (elem > group[foundindex].maxElem)
                {
                    group[foundindex].maxElem = elem;
                }
            }
            else // создаём новую //
            {
                GroupInfo< S,C,B> newgroup;
                newgroup.key = curKey;
                newgroup.count =1;
                newgroup.maxElem = elem;
                newgroup.minElem = elem;
                newgroup.sum = curVal;
                newgroup.avg = 0.0;
                group.push_back(newgroup);
            }
        }

        for (int i =0;i<group.size();i++)
        {
            group[i].avg = group[i].sum / group[i].count;

        }
    return group;
    }
}

int main()
{
    int arrInt[] = {12 ,202, 2 , 5,15,7};
    vector<GroupInfo<int,int,int>> resInt = stats<int,int,int>
(arrInt,6,IntKeyFunctor(),IntValueFunc());

    cout << "    For Int    " << endl;
    for (int i = 0; i < resInt.size(); i++)
```

```
    {
        cout << "Key: " << resInt[i].key
            << "  Count: " << resInt[i].count
            << "  Min: " << resInt[i].minElem
            << "  Max: " << resInt[i].maxElem
            << "  Sum: " << resInt[i].sum
            << "  Avg: " << resInt[i].avg << endl;
    }
    double arrDouble[] = {-5.5,3.2, -7.8,12.5, 25.6};
    vector<GroupInfo<double, int, double>> resDouble =
stats<double,int,double>(arrDouble,5,DoubleKeyFunctor(),DoubleValu
eFunc());

    cout << endl << "    For Double     " << endl;
        for (int i = 0; i < resDouble.size(); i++)
        {
            cout << "Key: " << resDouble[i].key
                << "  Count: " << resDouble[i].count
                << "  Min: " << resDouble[i].minElem
                << "  Max: " << resDouble[i].maxElem
                << "  Sum: " << resDouble[i].sum
                << "  Avg: " << resDouble[i].avg << endl;
        }

    string arrStr[] = {"apple","banana",
"august","chery","borticiii"};
    vector<GroupInfo<string, char, int>> resString =
stats<string,char,int>(arrStr,5,StringKeyFunctor(),StringValueFunc
());
    cout << endl << "    For String     " << endl;
        for (int i = 0; i < resString.size(); i++)
        {
            cout << "Key: " << resString[i].key
                << "  Count: " << resString[i].count
                << "  Min: " << resString[i].minElem
                << "  Max: " << resString[i].maxElem
                << "  Sum: " << resString[i].sum
                << "  Avg: " << resString[i].avg << endl;
        }
    return 0;
}
```

4. Результаты работы программы Программа корректно обрабатывает массивы типов int, double и string, формируя группы и выводя статистику по каждой из них.