# Инфа. ДЗ по шаблонам функций.

20 марта 2021 г.

№1.

Перегрузить оператор вывода для вектора, мапа и пары (pair); в перегрузке для мапа использовать уже готовую перегрузку пары. Но чтобы прога могла вывести и вектор, мап и пару интов, и даблов, и стрингов, итд. Сделать так, чтобы в каждой из 3 функций перегрузки (для вектора, мапа и пары) была только 1 строчка return os«... (максимально ликвидируем повторяющийся код). Достичь этого можно с помощью создания функции to_string, принимающей на вход контейнер, кладущий каким-то методом (например, stringstream) его в строку, и возвращающей это строку, которая и будет выведена в os. Т.е. в функции перегрузки будет что-то вроде return os«...to_string(collection)... Можно в функцию to_string подать в качестве второго аргумента еще символ-разделитель, ведь мы по-разному будем оформлять вывод вектора, мапа и пары.

№2.

```
#include <iostream>
#include <fstream>
#include <map>
#include <string>
#include <set>
#include <iterator>
#include <algorithm>
#include <iomanip>

using namespace std;

/*заметьте, я по 2 функции f1 и f2 написал, одна для вектора
 * вторая для мапа
```

```
 */

bool f1 (..)/*функция, проверяющая, есть ли элемент в данном
 векторе. true, если есть*/

bool f1 (..)/*функция, проверяющая, есть ли элемент в данном
 мапе. true, если есть*/

bool f2 (..)/*функция, проверяющая, есть ли в данном
 векторе четное число данных. true, если это правда*/

bool f2 (..)/*функция, проверяющая, есть ли в данном
 мапе четное число данных. true, если это правда*/

class checker {
public:
        int counter = 0;/*счетчик, сколько раз функция для вектора
        контейнеров вернула true*/


        void check(foo, vector<container>, element) {/*3 шаблонных
            * параметра: функция, которую вызываем, элемент, который
            * ищем в контейнере, и сам
            * контейнер. т.е. должна жрать и вектор, и мап.
                * Вызвать функцию foo для каждого контейнера
                * из вектора контейнеров, проверить, есть ли в
                * нем элемент. если есть, counter++*/
        }

        ~checker {
                //деструктор; вывести на экран counter
        }
};


int main() {
        /*объявить объект класса checker; инициализировать вектор
         *  мапов и вектор векторов, скормить их функции check:
```

```
         *  check(f1, map,..); check(f1, vector,..);
         *  check(f2, map,..); check(f2, vector,..);
         *  когда прога закончится, она в силу деструктора
         *  класса checker выведет на экран, для скольки
         *  элементов функция сработала
         */
        return 0;
}

/*если возникнет проблема с передачей параметра в шаблонную функцию,
 * а оттуда - в другую шаблонную функцию, то см. пример ниже
 * как передавать
 */

template<typename T>
void func(T a) {
        cout<<"kaban"<<endl;
}

template<class foo, typename T>
void f(foo f_, T param) {
        cout<<"Меня зовут"<<endl;
        f_(1);
}


int main() {
        f(func<int>, 1);/*
        * второй параметр (1) просто так, по приколу, чтобы показать,
        * как передавать с другими параметрами
        * */
        return 0;
}
```