**7 kyu**

**Sub-array elements sum**

4282% of 6923 of253[GiacomoSorbi](https://www.codewars.com/users/GiacomoSorbi)

C++

* [TRAIN AGAIN](https://www.codewars.com/kata/sub-array-elements-sum/train/cpp)
* [NEXT KATA](https://www.codewars.com/trainer/cpp)

Details

[Solutions](https://www.codewars.com/kata/sub-array-elements-sum/solutions/cpp)

[Forks (1)](https://www.codewars.com/kata/sub-array-elements-sum/forks/cpp)

[Discourse (34)](https://www.codewars.com/kata/sub-array-elements-sum/discuss/cpp)

* Add to Collection
* |
* Share this kata:

Given an array (or list or vector) of arrays (or, guess what, lists or vectors) of integers, your goal is to return the sum of a specific set of numbers, starting with elements whose position is equal to the main array length and going down by one at each step.

Say for example the parent array (etc, etc) has 3 sub-arrays inside: you should consider the third element of the first sub-array, the second of the second array and the first element in the third one: [[3, 2, 1, 0], [4, 6, 5, 3, 2], [9, 8, 7, 4]] would have you considering 1 for [3, 2, 1, 0], 6 for [4, 6, 5, 3, 2] and 9 for [9, 8, 7, 4], which sums up to 16.

One small note is that not always each sub-array will have enough elements, in which case you should then use a default value (if provided) or 0 (if not provided), as shown in the test cases.

<https://www.codewars.com/kata/sub-array-elements-sum/cpp>

#include <vector>

#include <iostream>

#include <stdio.h>

using namespace std;

long elementsSum(const std::vector<std::vector<int> >& arr, int d = 0){

  //your code here

    long sum = 0;

    int indice = arr.size() - 1;

    for(int i = 0; i < arr.size(); i++) {

        if(arr[i].size() > 0 &&  indice >= 0 && indice  < arr[i].size()) {

            sum += arr[i][indice];

        }

        else {

            sum += d;

        }

        indice--;

    }

    return sum ;

}

int main() {

    std::vector<std::vector<int> > v;

    std::vector<int> aux1;

    aux1.push\_back(-5);

    aux1.push\_back(-7);

    aux1.push\_back(-9);

    std::vector<int> aux2;

    aux2.push\_back(-90);

    aux2.push\_back(-22);

    aux2.push\_back(-10);

    v.push\_back(aux1);

    v.push\_back(aux2);

    cout << elementsSum(v, 1) << endl;

    return 0;

}