# ЦЕЛЬ РАБОТЫ

Экспериментальная проверка теоретически полученной функции трудоемкости для алгоритма точного решения задачи о сумме методом полного перебора. Получение практических навыков расстановки счетчика операций в программе на языке высокого уровня.

# ВАРИАНТ ЗАДАНИЯ

* Составить структурную схему алгоритма точного решения задачи о сумме;
* включить в функцию поиска точного решения задачи о сумме строки счетчика элементарных операций;
* для значений заданных вариантом задания №4 (таблица 1) использовать алгоритм точного решения задачи о сумме. Рассмотреть два варианта массива:

1. заполненный вручную;
2. заполненный с помощью генератора случайных чисел;

Таблица 1.

|  |  |  |
| --- | --- | --- |
| Размерность вектора случайных чисел | Максимальное случайное число в векторе | Значение суммы (V) |
| 13 | 90 | 25 |

# СТРУКТУРНАЫЕ СХЕМЫ АЛГОРИТМОВ



Рисунок 1 – Структурная схема алгоритма заполнения массива псевдослучайными элементами



Рисунок 2 – Структурная схема алгоритма вывода массива на экран



Рисунок 3 – Структурная схема алгоритма точного решения задачи о сумме

# ТЕКСТ ПРОГРАММЫ

#include <iostream>

#include <vector>

#include <cstdlib>

#include <cmath>

#include <ctime>

bool TaskSum(std::vector<int> S, size\_t N, int V, std::vector<int> &Cnt, size\_t &elOpCounter) {

bool fl = false; // 1(<-)

int i = 0; // 1(<-)

elOpCounter += 2;

size\_t lastIndex = N - 1; // 1(-)+1(<-)

elOpCounter += 2;

Cnt[lastIndex] = 1; // 1(lastIndex)+1(<-)

elOpCounter += 2;

do {

int Sum = 0; // 1(<-)

i = 0; // 1(<-)

elOpCounter += 2;

do {

Sum = Sum + S[i] \* Cnt[lastIndex - i]; // 1(<-)+1(+)+2(i)+1(\*)+1(-)

i = i + 1; // 1(<-) + 1(+)

} while (i < N); // 1(<)

// 8\*N

elOpCounter += 9 \* N;

if (Sum == V) { // 1(==)

fl = true; // 1(<-)

++elOpCounter;

return fl;

}

++elOpCounter;

int j = lastIndex; // 1(<-)

++elOpCounter;

while (Cnt[j] == 1) { // 1(j)+1(==)

Cnt[j] = 0; // 1(j)+1(<-)

j = j - 1; // 1(<-) + 1(-)

// 4

elOpCounter += 4;

}

elOpCounter += (N - j) \* 2;

Cnt[j] = 1; // 1(j)+1(<-)

elOpCounter += 2;

elOpCounter += 2;

} while (Cnt[0] != 1); //1(0)+1(!=)

return fl;

}

std::vector<int> randomFill(std::vector<int> arr) {

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

arr[i] = rand() % 90;}

return arr;

}

template <class T>

void printArr(std::vector<T> arr) {

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

std::cout << arr[i] << " ";

}

}

void clearTheArray(std::vector<int> &arr) {

size\_t i = 0;

do { arr[i] = 0;

i = i + 1;

} while (i < arr.size());

}

int main() {

const size\_t ARR\_SIZE = 13; // N

const int V = 25;

//RANDOM ARR CHECK

srand(time(0));

std::vector<int> mainRandArr(ARR\_SIZE);

std::vector<int> Counter(ARR\_SIZE);

mainRandArr = randomFill(mainRandArr);

std::cout << "Random array: " << std::endl;

printArr(mainRandArr);

std::cout << std::endl;

size\_t elOpCounter = 0;

TaskSum(mainRandArr, ARR\_SIZE, V, Counter, elOpCounter);

printArr(Counter);

std::cout << std::endl;

std::cout << "Elementary Operations Amount: " << elOpCounter\*2 << std::endl;

int Hn = (8\*ARR\_SIZE\*pow(2,ARR\_SIZE)+16\*pow(2,ARR\_SIZE)-3\*ARR\_SIZE-12);

std::cout << "F^(N): " << Hn << std::endl;

//CUSTOM ARR CHECK

std::vector<int> mainCustomArr(ARR\_SIZE);

for(size\_t i = 0; i < mainCustomArr.size(); ++i){

std::cin >> mainCustomArr[i];

}

std::cout << std::endl << "Custom array: " << std::endl;

printArr(mainCustomArr);

std::cout << std::endl;

elOpCounter = 0;

clearTheArray(Counter);

TaskSum(mainCustomArr, ARR\_SIZE, V, Counter, elOpCounter);

printArr(Counter);

std::cout << std::endl;

std::cout << "Elementary Operations Amount: " << elOpCounter\*2 << std::endl;

return 0;

}

# РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

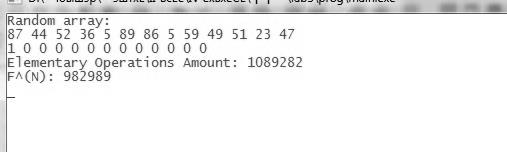


Рисунок 4 – Результат выполнения программы на псевдослучайном наборе входных данных

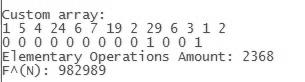


Рисунок 5 – Результат выполнения программы при пользовательском наборе входных данных

# СРАВНЕНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Для теоретического расчета трудоемкости алгоритма, была использована формула нахождения детальной оценки для худшего случая, используя методику последовательного подсчета операций:

(1)

В результате эксперимента было выявлено, что теоретические значение, полученные с помощью формулы (1), приближенно совпадают со значениями, полученными при полном переборе значений, с входными данными, полученными с помощью генератора случайных чисел. На рисунке 4 видно, что при полном переборе значений массива разница между теоретически и практически расчетами функции трудоемкости (рис.4) составляет 9,8%, что вполне приемлемо для массивов малой размерности.

Стоить заметить, что в лучшем случае, когда последний элемент массива совпадает с искомым значением, необходимо выполнить только одно суммирование, что приводит к оценке:

(2)

# ВЫВОДЫ

В ходе лабораторной работы был реализован алгоритм точного решения задачи о сумме на языке C++ с расстановкой счетчиков подсчета элементарных операций., а так же проведен сравнительный анализ функций трудоемкости полученных теоретическим и экспериментальным путем, для данного алгоритма. На основании анализа был сделан вывод, что полученные результаты приближенно совпадают, так как расхождение практической и теоретической оценок составляет всего 9,8%.