

Федеральное государственное автономное  
образовательное учреждение  
высшего образования  
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ  
ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

**ОТЧЕТ**  
по дисциплине Алгоритмы и структуры данных  
Практическая работа №2 — Линейные структуры данных

Преподаватель

подпись, дата

**Матковский И. В.**

инициалы, фамилия

Студент

**КИ19-07Б, 031941597**

номер группы, зачётной книжки

подпись, дата

**Горбацевич А. А.**

инициалы, фамилия

Красноярск 2020

## Содержание

1. Задание на работу.....	3
1.1 Разработать для решения поставленной задачи алгоритм; реализовать полученный алгоритм с использованием линейных структур данных заданных типов. Одна из заданных структур должна быть реализована самостоятельно, без использования готовых библиотек; вторая структура может быть как реализована самостоятельно, так и взята из STL.....	3
2. Задание на вариант.....	4
2.1 Задана арифметическая прогрессия с одним пропущенным элементом. Выявить пропущенное и восстановить его в положенном месте.....	4
3. Исходный код программы.....	5
4. Теоретические оценки временной сложности алгоритмов.....	9
4.1 Алгоритм с собственной линейной структуры.....	9
4.2 Алгоритм с использованием структуры vector из STL.....	9
4.3 Пояснение.....	9
5. Экспериментальные оценки временной и пространственной сложности программы.....	10
Приложение А Результаты работы программы.....	11

## **1. Задание на работу**

1.1 Разработать для решения поставленной задачи алгоритм; реализовать полученный алгоритм с использованием линейных структур данных заданных типов. Одна из заданных структур должна быть реализована самостоятельно, без использования готовых библиотек; вторая структура может быть как реализована самостоятельно, так и взята из STL.

## **2. Задание на вариант**

2.1 Задана арифметическая прогрессия с одним пропущенным элементом.  
Выявить пропущенное и восстановить его в положенном месте.

### 3. Исходный код программы

```
// dsaa_02.cpp
// Горбацевич Андрей
#include <iostream>
#include <map>
#include <vector>
#include <chrono>
#include <cmath>
#include <cassert>

#ifndef DEBUG
#define printf_d printf
#else
#define printf_d(...)
#endif

class DynIntArr { // ngl, can be done better
private:
    int *container;
    int _size = 0;
    int _capacity;

    void _resize(int new_size) {
        int *new_container = new int [new_size];
        for (int i = 0; i < this->_size; i++) {
            *(new_container + i) = *(this->container + i);
        }
        std::swap(this->container, new_container);
        delete [] new_container;

        this->_capacity = new_size;
    }
public:
    explicit DynIntArr(int initial_size=8) {
        this->_capacity = initial_size;
        this->container = new int [this->_capacity]{0};
    }
    ~DynIntArr() {
        if (this->container != nullptr) {
            delete [] this->container;
        }
    }

    int &at(int pos) {
        if (pos < 0 || pos > this->_capacity) {
            throw std::runtime_error("Illegal array access");
        }
        return *(this->container + pos);
    }

    void add(int val) {
        if (this->_size+1 > this->_capacity) {
            this->_resize(this->_capacity*2);
        }
        this->at(this->_size++) = val;
    }
}
```

```

int size() const { // NOLINT(modernize-use-nodiscard)
    return this->_size;
}

int capacity() const { // NOLINT(modernize-use-nodiscard)
    return this->_capacity;
}
};

inline void time_passed(std::chrono::system_clock::time_point start, double &holder) {
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(stop - start);
    holder = duration.count();
}

void algo_01(DynIntArr &cont); // algo with my container
void algo_02(std::vector<int> &cont); // algo with stl's vector

// let's assume we are not including reading of data into algo's time
// 'cause it can at worst be O(inf)
int main() {
    int reserve = 50;
    DynIntArr dyn_arr(reserve);
    std::vector<int> vec_int;
    vec_int.reserve(reserve);

    int N = 0;
    std::cout << "N=";
    std::cin >> N;
    for (int i = 0; i < N; i++) {
        int num = 0;
        std::cin >> num;
        dyn_arr.add(num);
        vec_int.push_back(num);
    }

    double fft, sft = fft = 0;

    if (N <= 2) { // Well, even if smth. is missed there is no way to determine
        goto endo;
    }

    { // algo with my container
        auto start = std::chrono::high_resolution_clock::now();
        algo_01(dyn_arr);
        time_passed(start, fft);
    }
    { // algo with stl's vector
        auto start = std::chrono::high_resolution_clock::now();
        algo_02(vec_int);
        time_passed(start, sft);
    }
    { // let's check integrity of content
        assert(dyn_arr.size() == vec_int.size());
        for (int i = 0; i < dyn_arr.size(); i++) {
    }
}

```

```

        assert(dyn_arr.at(i) == vec_int.at(i));
    }
}

endo:

std::cout << "Fixed list: ";
for (int i = 0; i < vec_int.size(); i++) {
    std::cout << vec_int.at(i) << " ";
}
std::cout << std::endl;

printf(
    "algo_01 is %s than algo_02 by %.0f microseconds",
    (fft > sft? "slower" : "faster"), fabs(fft - sft)
);

return 0;
}

void algo_01(DynIntArr &cont) {
    if (cont.size() <= 2) { // See this_file:97
        return;
    }

    int step = (cont.at(cont.size() - 1) - cont.at(0)) / (cont.size() - 1);
    int miss_pos = -1;
    for (int i = 1; i < cont.size(); i++) {
        int p = cont.at(i - 1),
            c = cont.at(i);

        if (c - p != step) {
            miss_pos = i;
            break;
        }
    }

    if (miss_pos != -1) {
        int val = cont.at(0) - step;
        int elc = cont.size() + 1;
        cont = DynIntArr(cont.size() + 1);
        for (int i = 0; i < elc; i++) {
            cont.add(val += step);
        }
    }
}

void algo_02(std::vector<int> &cont) {
    if (cont.size() <= 2) { // See this_file:97
        return;
    }

    int step = (cont.at(cont.size() - 1) - cont.at(0)) / (cont.size() - 1);
    int miss_pos = -1;
    for (int i = 1; i < cont.size(); i++) {
        int p = cont.at(i - 1),
            c = cont.at(i);
    }
}

```

```
if (c - p != step) {
    miss_pos = i;
    break;
}

if (miss_pos != -1) {
    int val = cont.at(0) - step;
    int elc = cont.size() + 1;
    cont.clear();
    for (int i = 0; i < elc; i++) {
        cont.push_back(val += step);
    }
}
```

// Задана арифметическая прогрессия с одним пропущенным элементом.  
// Выявить пропущенное и восстановить его в положенном месте.

## **4. Теоретические оценки временной сложности алгоритмов**

4.1 Алгоритм с собственной линейной структуры

Временная сложность алгоритма:  $O(n)$ .

4.2 Алгоритм с использованием структуры vector из STL

Временная сложность алгоритма:  $O(n)$ .

4.3 Пояснение

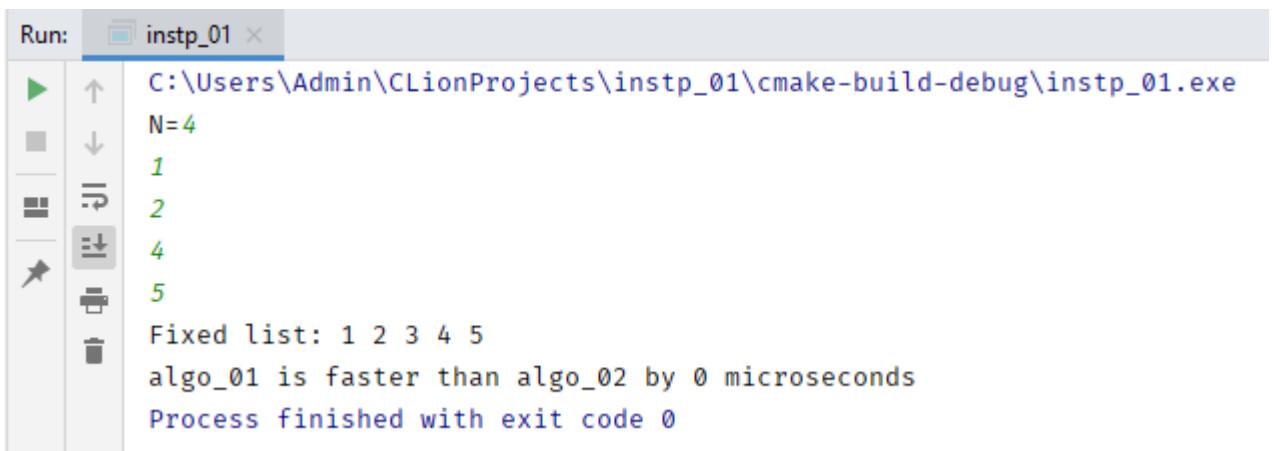
В данном случае разница проявилась в использовании памяти: реализованная структура была сжата в процессе работы алгоритма, поэтому её потребление памяти было меньше, чем у vector;

## 5. Экспериментальные оценки временной и пространственной сложности программы

Итоговый размер контейнера	Время, algo_01, микросекунды	Занимаемое пространство, algo_01, байты	Время, algo_02, микросекунды	Занимаемое пространство, algo_02, байты
5	0	80	0	80
4	0	20	0	80
4	0	20	0	80
4	0	20	0	80
25	0	44	0	80
25	0	44	0	80
25	0	48	0	80
25	0	48	0	80

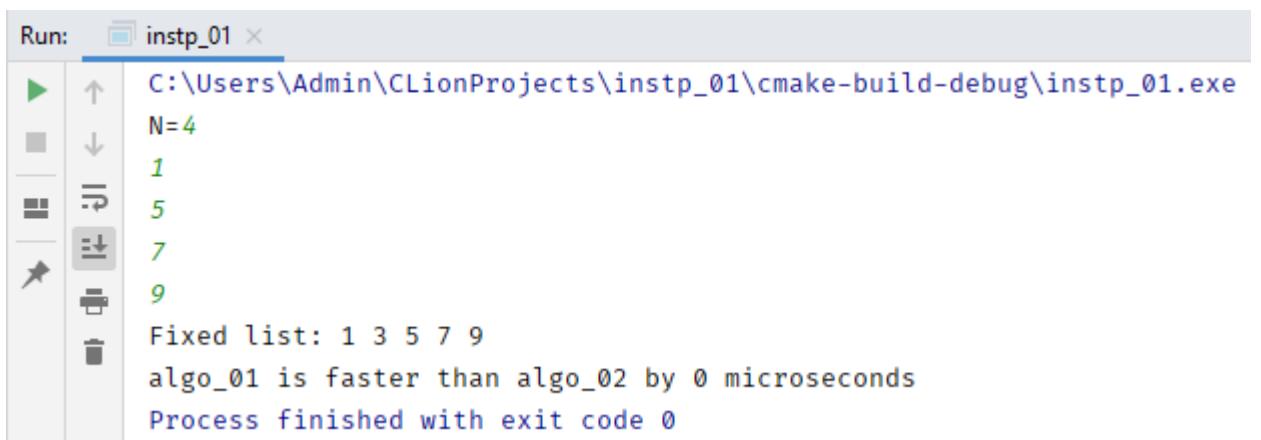
## Приложение А

### Результаты работы программы



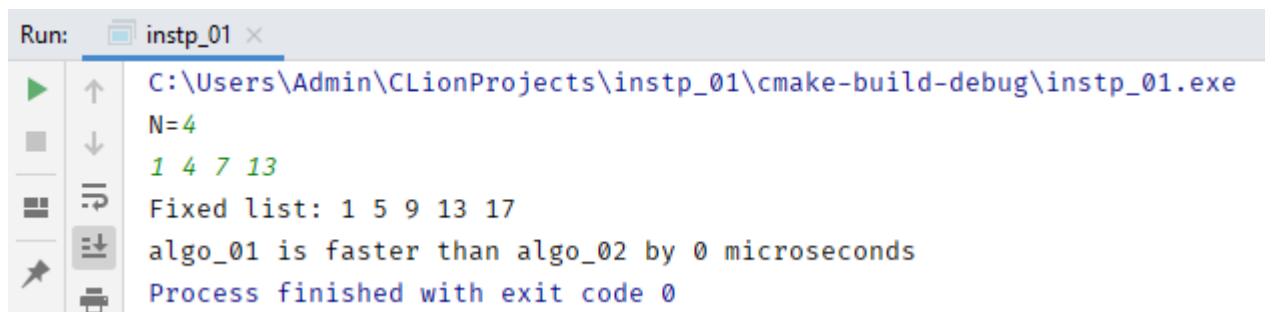
```
Run: instp_01 ×
C:\Users\Admin\CLionProjects\instp_01\cmake-build-debug\instp_01.exe
N=4
1
2
4
5
Fixed list: 1 2 3 4 5
algo_01 is faster than algo_02 by 0 microseconds
Process finished with exit code 0
```

Рисунок 1: Результат работы программы, пропуск в середине



```
Run: instp_01 ×
C:\Users\Admin\CLionProjects\instp_01\cmake-build-debug\instp_01.exe
N=4
1
5
7
9
Fixed list: 1 3 5 7 9
algo_01 is faster than algo_02 by 0 microseconds
Process finished with exit code 0
```

Рисунок 2: Результат работы программы, пропуск в начале



```
Run: instp_01 ×
C:\Users\Admin\CLionProjects\instp_01\cmake-build-debug\instp_01.exe
N=4
1 4 7 13
Fixed list: 1 5 9 13 17
algo_01 is faster than algo_02 by 0 microseconds
Process finished with exit code 0
```

Рисунок 3: Результат работы программы, пропуск в конце