ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | М. А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Анализ сложности алгоритмов |
| по курсу: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы**

Целью работы является изучение методов и получение практических навыков анализа сложности алгоритмов.

**1.2 Задание на лабораторную работу**

Используя память, пропорциональную n, хранить массив целых чисел A, содержащий n элементов. Элементы массива A могут принимать случайные значения от -((n div 2) - 1) до (n div 2). То есть, если в массиве хранится 10 элементов, то эти элементы должны быть в диапазоне от -4 до 5. Разработать алгоритм, который осуществляет заполнение массива A случайными значениями, и по выбору пользователя выполняет одну из двух функций. Состав выполняемых функций и требования к теоретической временной сложности этих функций определяется вариантом задания. Варианты задания приведены в таблице 1.

**Вариант 17**



**Ход решения**

**Листинг**

#include <random>

#include <iostream>

#include <string>

using namespace std;

double fill() {

std::string input;

while (true) //проверка введенного числа

{

bool error = 0;

std::cin >> input;

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

if ((isdigit(input[i]) == 0 && input[i] != '.' && input[i] != '-')) {

error = 1;

break;

}

}

if (error == 1) {

printf("Enter value without letters\n");

}

else

{

break;

}

}

return stoi(input);

}

int\* gen\_array(int n)

{

random\_device rd;

mt19937 gen(rd());

int\* arr = new int[n];

for (int i = 0; i < n; i++)

{

int rand\_num = gen();

int pos = gen()%2;

if (pos == 1)

{

arr[i] = rand\_num % 6;

}

else

{

arr[i] = -rand\_num % 4-1;

}

}

return arr;

}

void show\_array(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

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

}

cout << endl;

}

int\* decrease\_by\_two(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

arr[i] = arr[i] - 2;

}

return arr;

}

int count\_pos(int arr[], int n)

{

int num\_of\_pos = 0;

for (int i = 0; i < n; i++)

{

if (arr[i] >= 0)

{

num\_of\_pos += 1;

}

}

return num\_of\_pos;

}

int main()

{

cout << "Enter len "<<endl;

int n = fill();

int\* arr = gen\_array(n);

cout << "Your array: ";

show\_array(arr, n);

cout << "Enter:\n1 if you want to decrease all elementts by 2\n2 if you want to count positive elements\n";

while (1) {

int input = fill();

;

if (input == 1)

{

decrease\_by\_two(arr, n);

cout << "Decreased array: ";

show\_array(arr, n);

cout << endl;

}

else if (input == 2) {

cout << "Number of positive elements: "<<count\_pos(arr, n);

cout << endl;

}

else {

break;

}

}

}

**Подсчет теоретических сложностей**

Алгоритм использует следующие данные (если не учитывать проверку введенных значений):

-4 переменных целого типа

-1 целочисленный массив размера n (ввод с клавиатуры)

**Пространственная сложность определяется следующей формулой:**

**= n\*4 + 8**

Где Cint – объем целочисленной переменной

**Теоретическая пространственная сложность составляет**

V(n) = O(v) = O(max(O(n\*Cint), O(2\*Cint))) = O(max(O(n), O(1))) = O(n)

**Временная сложность**