ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Рогачев С. А. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| «АНАЛИЗ СЛОЖНОСТИ АЛГОРИТМОВ» |
| по курсу: Структуры и алгоритмы обработки данных |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М011 |  |  |  | Борисов С. И. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

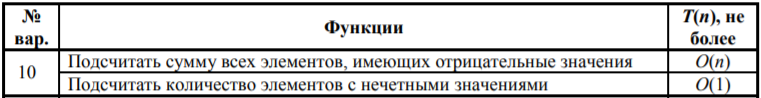
Санкт-Петербург 2021

**Цель работы**

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

**Задание на лабораторную**

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.



**Листинг программы, реализующей алгоритм**

// М011 ЛБ-1 Боривов С. И. Вариант 10.

#include <iostream>

using namespace std;

void NOT\_Chet(int x);

int Minus\_summ(int\* c, int y);

void NOT\_Chet(int x) // Функция считающая кол-во нечетных элементов в массиве.

{

cout << endl << "Кол-во нечетных элементов в массиве :" << x << "\n";

}

int Minus\_summ(int\* c, int y) // Функция считающая cумму отрицательных элементов в массиве.

{

int minusSumm = 0;

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

{

if (c[i] < 0) // если элемент массива меньше 0, то сумма отриц увеличивается.

minusSumm += c[i];

}

cout << endl << "Сумма отрицательных элементов в массиве :" << minusSumm << "\n";

return 0;

}

int main()

{

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите размер массива: ";

cin >> n;

int \*randomDigits =new int [n]; // создание динамического массива.

cout << "Заполнение массива случайными цифрами: \n";

srand(time(NULL));

for (int i = 0; i < n; i++) // заполнение массива случайными числами.

{

randomDigits[i] = 0 + rand() % n - ((n/2) - 1);

}

cout << endl << "Массив c цифрами: " << "\n";

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

{

cout << randomDigits[i] << " ";

}

int Key\_board = 0;

while (Key\_board == 0)

{

cout << endl << "Меню: \n";

cout << "Введите 1, если хотите подсчитать кол-во нечетных элементов в массиве. \n";

cout << "Введите 2, если хотите подсчитать сумму отрицательных элементов в массиве. \n";

cout << "Введите 3, если хотите выйти из программы. \n";

cout << "Ввод: ";

cin >> Key\_board;

switch (Key\_board)

{

case 1:

{

// NOT\_Chet(randomDigits, n);

int not\_chet\_summ = 0;

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

{

if ((randomDigits[i] % 2) != 0) // Если остаток от деления на 2 не равен 0, то сумма нечетных увеличиваеться на 1

not\_chet\_summ += 1;

}

NOT\_Chet(not\_chet\_summ);

int Key\_board\_2 = 0;

cout << "Введите 1, если хотите вернуться в меню \n";

cout << "Введите 2, если хотите выйти из программы. \n";

cout << "Ввод: ";

while (Key\_board\_2 == 0)

{

cin >> Key\_board\_2;

switch (Key\_board\_2)

{

case 1:

{

Key\_board = 0;

break;

}

case 2:

{

cout << "Осуществляеться выход из программы..." << endl;

exit(EXIT\_SUCCESS);

break;

}

default:

{

cout << "Ошибка! Запрос введен некоректно!";

cout << "Повторите ввод: ";

Key\_board\_2 = 0;

break;

}

}

}

break;

}

case 2:

{

Minus\_summ(randomDigits, n);

int Key\_board\_3 = 0;

cout << "Введите 1, если хотите вернуться в меню \n";

cout << "Введите 2, если хотите выйти из программы. \n";

cout << "Ввод: ";

while (Key\_board\_3 == 0)

{

cin >> Key\_board\_3;

switch (Key\_board\_3)

{

case 1:

{

Key\_board = 0;

break;

}

case 2:

{

cout << "Осуществляеться выход из программы..." << endl;

exit(EXIT\_SUCCESS);

break;

}

default:

{

cout << "Ошибка! Запрос введен некоректно!";

cout << "Повторите ввод: ";

Key\_board\_3 = 0;

break;

}

}

}

break;

}

case 3:

{

cout << "Осуществляеться выход из программы..." << endl;

exit(EXIT\_SUCCESS);

break;

}

default:

{

cout << "Ошибка! Запрос введен некоректно!";

cout << "Повторите ввод: ";

Key\_board = 0;

break;

}

}

}

delete[]randomDigits;

return 0;

}

**Расчет пространственной и временной сложностей алгоритма**

В данном алгоритме содержится, следующие переменные:

1. Один массив размерностью n;

2. 4 константы целого типа: Key\_board, Key\_board\_2, Key\_board\_3, Minus\_summ;

Таким образом, пространственная сложность алгоритма определяется следующим образом:

V = 4\*Cint+n\*Cint

где Cint – константа, характеризующая объем памяти, отводимый под переменную целого типа.

**Расчет теоретической пространственной и теоретической временной сложностей алгоритма**

Теоретическая пространственная сложность алгоритма составляет:

V(n) = O(n) = O(max(O(4\*Cint), O(n\*Cint)))= O(O(1),O(n)) = O(n)

Временную сложность алгоритма определяем на основе анализа текста программы, реализующей этот алгоритм. Согласно заданию на лабораторную работу, необходимо реализовать две функции, теоретические временные сложности которых не превышают заданных. Поэтому необходимо рассчитать теоретическую временную сложность функции, реализующих эти алгоритмы:

Tнечет = K11

Tотриц = K16 +n\*(K20+K21)

Tалгоритм = K31+K35+n\*K39+n\*K44+K65+n\*K67+K70+Tнечет+Tотриц

Теоретическая временная сложность функций составляет:

Tнечет =O(нечет) = O(max(O(K11))) = O(max(O(1)) = O(1)

Tотриц = O(Tотриц) = O(max(O(K16),O(n\*( K20+K21)))= O(max(O(1),O(n),O(1))) = O(n)

**Выводы**

На основе этих расчетов можно сделать вывод, что был разработан алгоритм, характеристики которого соответствуют поставленному заданию.