# ЦЕЛЬ РАБОТЫ

В лабораторной работе требуется провести экспериментальную проверку теоретической оценки трудоемкости алгоритма поиска минимума, а так же ознакомиться с принципами использования генератора случайных чисел для создания наборов исходных данных.

# ВАРИАНТ ЗАДАНИЯ

* Составить структурную схему и написать программу поиска минимума в массиве сгенерированных псевдослучайных чисел.
* Написать программу подсчета n-го гармонического числа.
* Подсчитать количество операций переприсваивания для программной реализации поиска минимума в массиве случайных чисел.
* Внести изменения в соответствующую программу. Длину массива и максимальное случайное число в последовательности соответствует варианту задания №4:

Таблица 1.

|  |  |
| --- | --- |
| Наибольшее случайное число в последовательности | Количество элементов в массиве случайных чисел |
| 950 | 250, 300, 400 |

# 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 поиска n-го гармонического числа



Рисунок 4 – Структурная схема алгоритма поиска минимального элемента и подсчета количества переприсваиваний

# ТЕКСТ ПРОГРАММЫ

# #include <iostream>

# #include <iomanip>

# #include <windows.h>

# #include <time.h>

# #include <cmath>

# using namespace std;

# struct result\_t{

# size\_t assigmentCounter = 0;

# size\_t minIndex = 0;

# };

# const size\_t ARR\_SIZE1 = 250,

# ARR\_SIZE2 = 300,

# ARR\_SIZE3 = 400;

# const int MAX\_VAL = 950;

# int \*randomArrFill(int \* const arr, const size\_t arrSize, const int MAX\_VAL){

# for (size\_t i = 0; i < arrSize; ++i){

# arr[i] = rand() % MAX\_VAL;

# }

# return arr;

# }

# void printArr(const int \* const arr, const size\_t arrSize){

# for (size\_t i = 0; i < arrSize; ++i){

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

# }

# }

# result\_t findMin(const int \* const arr, const size\_t arrSize){

# result\_t result;

# result.minIndex = 0;

# result.assigmentCounter = 0;

# for(size\_t i = 1; i < arrSize; ++i){

# if (arr[i] < arr[result.minIndex]){

# result.minIndex = i;

# ++result.assigmentCounter;

# }

# }

# return result;

# }

# double harmonicN(size\_t arrSize){

# double result = 0.0;

# for(size\_t i = 1; i < arrSize; ++i){

# result += 1.0/static\_cast<double>(i);

# }

# return result;

# }

# int main(){

# srand(time(0)); //randomize function

# int arr1[ARR\_SIZE1] = {0},

# arr2[ARR\_SIZE2] = {0},

# arr3[ARR\_SIZE3] = {0};

# 

# randomArrFill(arr1, ARR\_SIZE1, MAX\_VAL);

# randomArrFill(arr2, ARR\_SIZE2, MAX\_VAL);

# randomArrFill(arr3, ARR\_SIZE3, MAX\_VAL);

# double Hn1 = 0.0,

# Hn2 = 0.0,

# Hn3 = 0.0;

# 

# Hn1 = harmonicN(ARR\_SIZE1);

# Hn2 = harmonicN(ARR\_SIZE2);

# Hn3 = harmonicN(ARR\_SIZE3);

# 

# result\_t test1, test2, test3;

# 

# test1 = findMin(arr1, ARR\_SIZE1);

# test2 = findMin(arr2, ARR\_SIZE2);

# test3 = findMin(arr3, ARR\_SIZE3);

# 

# cout.precision(2);

# cout << "ARRAY SIZE |" << " MIN VALUE |" << " ASSIGMENT AMOUNT |" << " HARMONIC" << endl;

# cout << setw(7) << ARR\_SIZE1 << setw(10) <<arr1[test1.minIndex] << setw(14) <<test1.assigmentCounter << setw(18) << Hn1 << endl;

# cout << setw(7) << ARR\_SIZE2 << setw(10) <<arr2[test2.minIndex] << setw(14) <<test2.assigmentCounter << setw(18) << Hn2 <<endl;

# cout << setw(7) << ARR\_SIZE3 << setw(10) <<arr3[test3.minIndex] << setw(14) <<test3.assigmentCounter << setw(18) << Hn3 <<endl;

# return 0;

# }

# РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

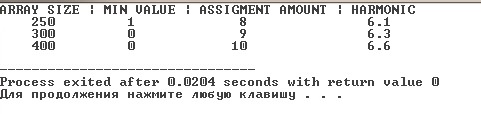


Рисунок 5 – Результат выполнения программы при первом запуске

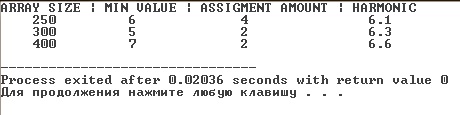


Рисунок 6 – Результат выполнения программы при втором запуске

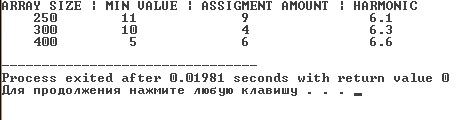


Рисунок 7 – Результат выполнения программы при третьем запуске

# СРАВНЕНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Для теоретического расчета среднего количества операций переприсваивания была использована формула :

(1)

В результате эксперимента было выявлено, что теоретические значение, полученные с помощью формулы (1), приближенно совпадают со значениями среднего количества переприсваиваний в программной реализации алгоритма поиска минимального значения. Полученные значения отображены в таблице 2.

Таблица 2.

|  |  |  |
| --- | --- | --- |
| Количество элементов массива | Кол-во переприсваиваний в программной реализации | n-ое гармоническое число |
| 250 | 6,06 | 6,1 |
| 300 | 6,25 | 6,3 |
| 400 | 6,5 | 6,6 |

Данные были получены при многократном тестировании программы. Практические результат отличается при конкретной реализации псевдослучайной последовательности.

# ВЫВОДЫ

В ходе лабораторной работы была написана программа для генерации массивов, размерами 250, 300 и 400 элементов, с псевдослучайными значениями, поиска минимального из этих элементов и подсчёта операций переприсваивания при этом. На основании экспериментов можно сделать вывод, что среднее значение количества операций переприсваивания в программной реализации поиска минимума в массиве псевдослучайных чисел, приближенно совпадает с теоретически найденным n-ым гармоническим числом.