Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 1

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы 609-21,

Шумилов И.Д.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

Сургут

2024 г.

**Цель работы**: освоить принципы формирования монотонных, частично упорядоченных и случайных последовательностей данных; изучить функции, позволяющие производить оценку длительности времени выполнения алгоритмов.

**Общее задание на лабораторную работу**

1. Разработать функции, формирующие упорядоченные (в том числе и в обратном порядке), частично упорядоченные и случайные последовательности целых чисел и чисел с плавающей запятой. При реализации функций считать, что выделение памяти под последовательности происходит вне этих функций, а в качестве формальных параметров функции получают указатель на массив, его размер и, возможно, диапазон изменения величин и длину интервалов (для частично упорядоченных последовательностей). Рекомендуется продумать единый прототип для функций, формирующих последовательности, определить его и использовать массив указателей на функции для автоматизации сбора статистической информации.

2. Оценить длительность формирования последовательностей всех типов для нескольких значений размеров последовательностей (, , …, ), и на основе полученных значений построить графики зависимостей длительностей формирования от размера последовательностей.

3. Составить отчет, в котором привести блок-схемы функций (возможно, на примере одной функции), реализующих формирование частично упорядоченных последовательностей, и главной функции, полученные графики зависимостей и выводы по полученным результатам.

**Блок-схемы алгоритмов функций формирования частично упорядоченных последовательностей**

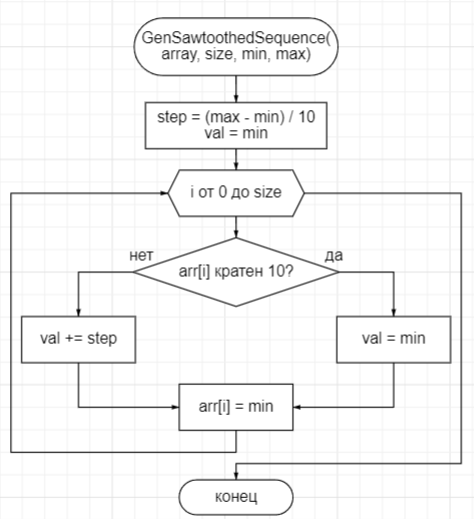


Рисунок 1. Блок-схема функции формирования зубчатой последовательности

**Блок-схема главных функций**

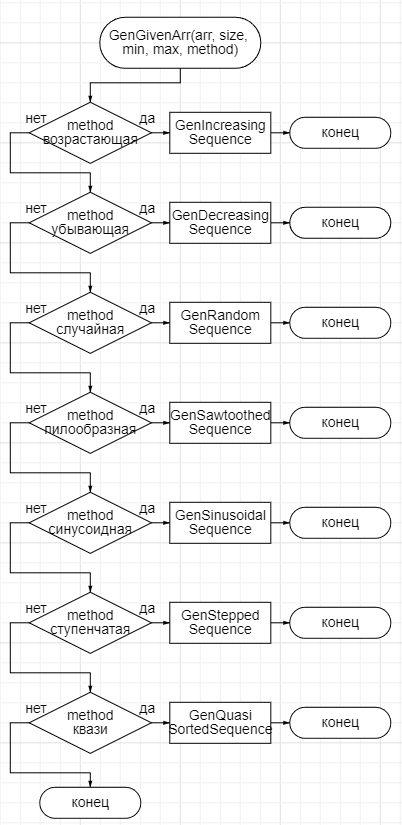


Рисунок 2. Блок-схема функции формирования последовательностей



Рисунок 3. Блок-схема функции засечения времени

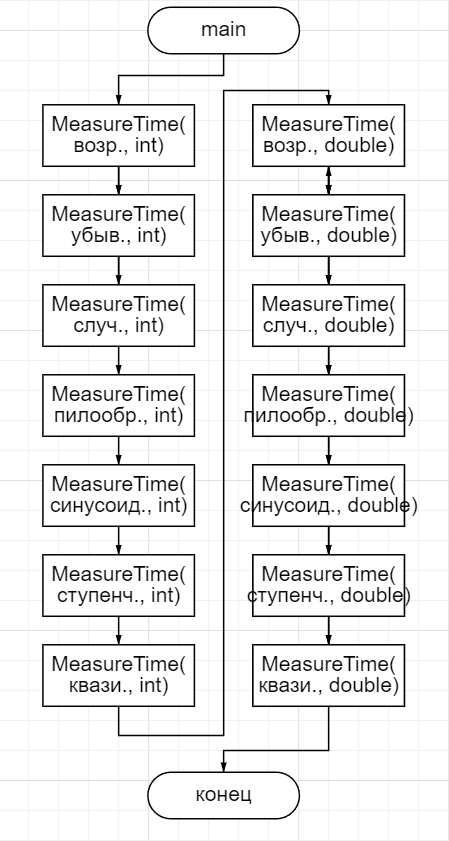


Рисунок 4. Блок-схема главной функции

**Листинг кода программы**

#include <iostream>

#include <math.h>

#include <chrono>

enum class Method { Increasing, Decreasing, Random, Sawtoothed, Sinusoidal, Stepped, QuasiSorted };

template<typename T>

T randV(T min, T max) {

double V = (double)(rand()) / RAND\_MAX;

return (T)(V \* (max - min + 1) + min);

}

template<typename T>

void GenIncreasingSequence(T \*arr, int size, T min, T max) {

T step = (max - min) / size;

T val = min;

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

arr[i] = val;

val += step;

}

}

template<typename T>

void GenDecreasingSequence(T \*arr, int size, T min, T max) {

T step = (max - min) / size;

T val = max;

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

arr[i] = val;

val -= step;

}

}

template<typename T>

void GenRandomSequence(T \*arr, int size, T min, T max) {

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

arr[i] = randV(min, max);

}

}

template<typename T>

void GenSawtoothedSequence(T \*arr, int size, T min, T max) {

T step = (max - min) / 10; // tooth interval is 10

T val = min;

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

if (i % 10 == 0)

val = min;

else

val += step;

arr[i] = val;

}

}

template<typename T>

void GenSinusoidalSequence(T \*arr, int size, T min, T max) {

// f(x) = amplitude \* sin(2 \* PI \* frequency \* x + shift)

double pi = atan(1) \* 4;

for (int i = 1; i <= size; i++) {

arr[i - 1] = (T)(100 \* sin(2 \* pi \* 0.05 \* i + 500));

}

}

template<typename T>

void GenSteppedSequence(T \*arr, int size, T min, T max) {

T step = (max - min) / (size / 10);

T val = min;

T error = step / 6;

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

if (i % 10 == 0 && i != 0) {

val += step;

}

arr[i] = randV(val - error < min ? min : val - error, val + error > max ? max : val + error);

}

}

template<typename T>

void GenQuasiSequence(T \*arr, int size, T min, T max) {

T step = (max - min) / size;

T val = min;

T error = step \* 2;

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

arr[i] = randV(val - error < min ? min : val - error , val + error > max ? max : val + error);

val += step;

}

}

template<typename T>

void PrintArr(T \*arr, int size) {

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

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

}

std::cout << "\b\b." << std::endl;

}

template<typename T>

void GenGivenArr(T \*arr, int size, T min, T max, Method m) {

switch (m)

{

case Method::Increasing:

GenIncreasingSequence(arr, size, min, max);

break;

case Method::Decreasing:

GenDecreasingSequence(arr, size, min, max);

break;

case Method::Random:

GenRandomSequence(arr, size, min, max);

break;

case Method::Sawtoothed:

GenSawtoothedSequence(arr, size, min, max);

break;

case Method::Sinusoidal:

GenSinusoidalSequence(arr, size, min, max);

break;

case Method::Stepped:

GenSteppedSequence(arr, size, min, max);

break;

case Method::QuasiSorted:

GenQuasiSequence(arr, size, min, max);

break;

}

}

template<typename T>

void MeasureTime(Method m, T min, T max) {

for (int i = 5; i <= 50; i += 5) {

T\* arr = (T\*)malloc(sizeof(T) \* i \* pow(10, 6));

auto start = std::chrono::high\_resolution\_clock::now();

GenGivenArr(arr, i, min, max, m);

auto end = std::chrono::high\_resolution\_clock::now();

std::chrono::duration<double, std::milli> time = end - start;

std::cout << time.count() << std::endl;

if (arr)

delete arr;

}

std::cout << std::endl;

}

int main(int argc, char\*\* argv) {

srand((unsigned)time(NULL));

std::cout << "Int\n" << std::endl;

MeasureTime(Method::Increasing, 100, 1000000);

MeasureTime(Method::Decreasing, 100, 1000000);

MeasureTime(Method::Random, 100, 1000000);

MeasureTime(Method::Sawtoothed, 100, 1000000);

MeasureTime(Method::Sinusoidal, 100, 1000000);

MeasureTime(Method::Stepped, 100, 1000000);

MeasureTime(Method::QuasiSorted, 100, 1000000);

std::cout << "Double\n" << std::endl;

MeasureTime(Method::Increasing, pow(10, 2), pow(10, 6));

MeasureTime(Method::Decreasing, pow(10, 2), pow(10, 6));

MeasureTime(Method::Random, pow(10, 2), pow(10, 6));

MeasureTime(Method::Sawtoothed, pow(10, 2), pow(10, 6));

MeasureTime(Method::Sinusoidal, pow(10, 2), pow(10, 6));

MeasureTime(Method::Stepped, pow(10, 2), pow(10, 6));

MeasureTime(Method::QuasiSorted, pow(10, 2), pow(10, 6));

return 0;

}

**Таблицы с результатами измерений длительности формирования последовательностей**

Таблица 1. Результаты измерений длительности формирования последовательностей типа int

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Размер последовательности | Время формирования последовательности (мс) | | | | | | |
| Возраста-  ющая | Убыва-  ющая | Случай-  ная | Пилооб-  разная | Синусои-  дальная | Ступен-  чатая | Квази-  упорядоченная |
| 5000000 | 17.16 | 17.83 | 264.93 | 32.6529 | 46.6868 | 269.576 | 265.755 |
| 10000000 | 34.18 | 34.45 | 529.341 | 67.1213 | 91.0548 | 539.975 | 529.761 |
| 15000000 | 50.94 | 49.88 | 788.731 | 101.023 | 136.394 | 812.995 | 800.37 |
| 20000000 | 71.60 | 68.67 | 1086.2 | 135.909 | 182.041 | 1086.61 | 1058.85 |
| 25000000 | 96.12 | 99.55 | 1320.81 | 170.237 | 227.985 | 1347.89 | 1323.55 |
| 30000000 | 96.95 | 110.66 | 1600.52 | 206.783 | 272.714 | 1621.22 | 1602.95 |
| 35000000 | 98.02 | 122.13 | 1843.63 | 238.773 | 333.179 | 1888.7 | 1874.31 |
| 40000000 | 109.54 | 111.67 | 2117.66 | 275.596 | 363.482 | 2173.67 | 2120.08 |
| 45000000 | 145.77 | 169.57 | 2369.47 | 312.947 | 410.64 | 2433.36 | 2396.94 |
| 50000000 | 187.62 | 185.87 | 2630.84 | 346.811 | 457.911 | 2706.99 | 2655.69 |
| Сумма | 907.9 | 970.28 | 14552.13 | 1887.853 | 2522.088 | 14880.99 | 14628.26 |

Таблица 2. Результаты измерений длительности формирования последовательностей типа double

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Размер последовательности | Время формирования последовательности (мс) | | | | | | |
| Возраста-  ющая | Убыва-  ющая | Случай-  ная | Пилооб-  разная | Синусои-  дальная | Ступен-  чатая | Квази-  упорядоченная |
| 5000000 | 21.0338 | 20.784 | 297.755 | 27.896 | 43.8381 | 270.524 | 266.095 |
| 10000000 | 40.7884 | 42.2558 | 585.77 | 57.3301 | 88.0068 | 551.948 | 535.542 |
| 15000000 | 62.5143 | 61.2397 | 814.37 | 87.4412 | 131.512 | 811.286 | 801.132 |
| 20000000 | 82.5937 | 81.7294 | 1071.98 | 117.02 | 175.97 | 1094.91 | 1068.1 |
| 25000000 | 101.824 | 101.233 | 1357.56 | 146.666 | 220.377 | 1356.94 | 1334.05 |
| 30000000 | 120.873 | 124.88 | 1585.26 | 177.409 | 264.277 | 1618.28 | 1605.35 |
| 35000000 | 144.01 | 145.168 | 1844.99 | 207.888 | 308.952 | 1891.79 | 1866.35 |
| 40000000 | 162.837 | 159.909 | 2105.98 | 238.325 | 351.39 | 2159.14 | 2142.45 |
| 45000000 | 183.193 | 181.943 | 2376.71 | 270.139 | 395.418 | 2444.26 | 2414.08 |
| 50000000 | 203.637 | 200.459 | 2647.08 | 300.469 | 441.845 | 2700.17 | 2685.3 |
| Сумма | 1123.304 | 1119.601 | 14687.46 | 1630.583 | 2421.586 | 14899.25 | 14718.45 |

**Графики зависимости длительности формирования последовательности от размера последовательности**

Рис 5. График зависимости формирования последовательностей типа int

Рис 6. График зависимости формирования последовательностей типа double

Рис 7. График зависимости формирования некоторых последовательностей типа int и double

**Пример работы функций**

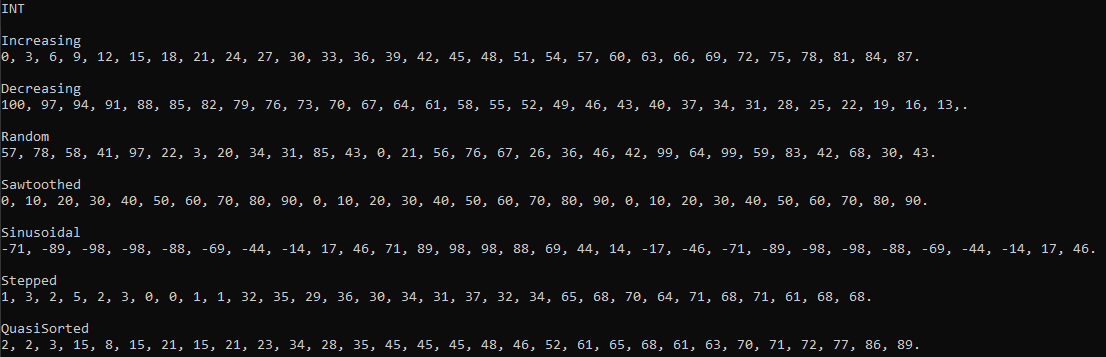
****

Рис 8. Демонстрация работы функций формирования последовательностей для типа int

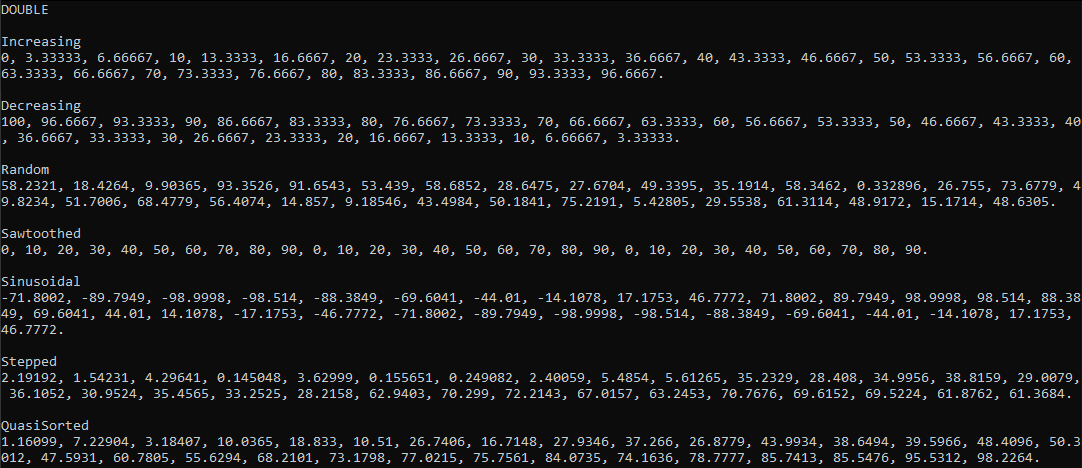
****

Рис 9. Демонстрация работы функций формирования последовательностей для типа double

**Выводы**

Были освоены принципы формирования монотонных, частично упорядоченных и случайных последовательностей данных; изучены функции, позволяющие производить оценку длительности времени выполнения алгоритмов. По суммарному времени формирования последовательностей для всех размеров был определен порядок последовательностей: убывающая, возрастающая, пилообразная, синусоидальная, квази-упорядоченная, ступенчатая, случайная. Длительность формирования у последовательностей типа int незначительно меньше, чем у последовательностей типа double.