Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

Кафедра Управления и интеллектуальных технологий

**Отчёт по лабораторной работе № 1**

# По курсу «Разработка ПО систем управления»

# «Основы языка С++»

# Выполнил студент группы А-03-19

# Мирошников А. М.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

2) Постановка задачи (Вариант 10) :

Отображайте гистограмму вертикально без подписей, например:

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*\*

\*\*

\*

Указание.

=========

Можно воспользоваться следующей логикой: проходить по всем столбцам и печатать \*,

если высота столбца больше номера строки, или пробел, если нет — и так до тех пор,

пока на очередной строке печатается хотя бы одна звездочка.

3) Логика решения варианта:

Использована логика из указания к индивидуальному заданию.

Внешний цикл - контроллирует номер строки для выбора печати " " или "\*" во внутреннем,

также - обеспечивает проход по всем строкам.

Внутренний цикл - цикл по столбцам. Производит масштабирование, если это необходимо,

печатает символы.

4) Исходный код всех модулей:

============

= main.cpp =

============

#include <iostream>

#include <vector>

#include <cmath>

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4;

using namespace std;

void find\_min\_max(vector<double> & numbers, double& min, double& max) {

min = numbers[0];

max = numbers[0];

for (double x : numbers) {

if (x < min) {

min = x;

}

else

if (x > max) {

max = x;

}

}

}

void sort\_nums\_to\_bins(vector<double> & numbers, vector<size\_t> & bins, double& min, double& max, size\_t& bin\_count, size\_t& max\_count) {

size\_t bin\_index;

max\_count = 0;

for (double number : numbers) {

bin\_index = (size\_t)( (number - min) \* bin\_count / (max - min) );

if (number == max) {

bin\_index--;

}

// abs for negatives ??

bins[bin\_index]++;

}

}

void print\_bins( vector<size\_t> & bins) {

for (size\_t bin: bins) {

cout << bin << "| ";

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

cout << "\*";

}

cout << "\n";

}

}

size\_t find\_max\_count( vector<size\_t> & bins) {

size\_t max = bins[0];

for (size\_t bin: bins) {

if (bin > max) {

max = bin;

}

}

return max;

}

void print\_bins\_scaled( vector<size\_t> & bins, size\_t & max\_count) {

// bool fl;

/\*

if (max\_count > MAX\_ASTERISK) {

fl = true;

} else {

fl = false;

}

\*/

max\_count = find\_max\_count(bins);

size\_t height;

double k = static\_cast<double>(MAX\_ASTERISK) / max\_count;

for (size\_t bin: bins) {

if ((bin) < 100) {

cout << " ";

}

if ((bin) < 10) {

cout << " ";

}

cout << bin << "|";

height = k \* bin;

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

cout << "\*";

}

cout << "\n";

}

}

void print\_bins\_vertical(vector<size\_t> & bins) {

size\_t max\_count = 0;

for (size\_t count : bins) {

if (count > max\_count) {

max\_count = count;

}

}

size\_t max\_height;

const bool scaling\_needed = max\_count > MAX\_ASTERISK;

double scaling\_factor = 1;

if (scaling\_needed) {

scaling\_factor = (double)MAX\_ASTERISK / max\_count;

max\_height = (size\_t)(max\_count \* scaling\_factor);

} else {

max\_height = max\_count;

}

for (size\_t i = 0; i < max\_height; i++) {

for (size\_t bin : bins) {

size\_t height = bin;

if (scaling\_needed) {

height = (size\_t)(bin \* scaling\_factor);

}

if (height > i) {

cout << "\*";

} else {

cout << " ";

}

}

cout << endl;

}

return;

}

int main()

{

// Input

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

vector<double> numbers(number\_count);

cerr << "Enter numbers:\n";

for (size\_t i = 0; i < number\_count; i++) {

cin >> numbers[i];

}

cerr << "Enter bin count: ";

size\_t bin\_count;

cin >> bin\_count;

// End of Input

vector<size\_t> bins(bin\_count, 0);

double min, max;

size\_t max\_count;

find\_min\_max(numbers, min, max);

sort\_nums\_to\_bins(numbers, bins, min, max, bin\_count, max\_count);

// print\_bins\_scaled(bins, max\_count);

print\_bins\_vertical(bins);

return 0;

}