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 № 3**

# По курсу «Разработка ПО систем управления»

# «Декомпозиция программы»

# Выполнил студент группы А-01-19

# Вареско С.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

**Задание.**

# 1) Написать программу для построения гистограммы массива чисел как изображения в формате SVG

2) Доработать программу в соответствии с вариантом.

#### Вариант 3

Измените высоту изображения IMAGE\_HEIGHT = 700. Дайте пользователю возможность задавать высоту столбца гистограммы. Если итоговая высота гистограммы больше IMAGE\_HEIGHT, рассчитывать высоту столбца как (IMAGE\_HEIGHT / bins\_count).

КОД ПРОГРАММЫ

***main.cpp***

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include "histogram.h" |
|  | #include "SVG.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | int main() |
|  | { |
|  | size\_t number\_count; |
|  | size\_t bin\_height; |
|  | cerr << "Enter number count: "; |
|  | cin >> number\_count; |
|  | if (number\_count==0) |
|  | return 1; |
|  | const auto numbers = input\_numbers(number\_count); |
|  | size\_t bin\_count; |
|  | cerr << "Enter column count: "; |
|  | cin >> bin\_count; |
|  | if (bin\_count==0) |
|  | return 1; |
|  | cerr << "Enter bin height: "; |
|  | cin >> bin\_height; |
|  | const auto bins = make\_histogram(numbers, bin\_count); |
|  | show\_histogram\_svg(bins, bin\_height, bin\_count); |
|  | return 0; |
|  | } |

***histogram.h***

|  |
| --- |
| #ifndef HISTOGRAM\_H\_INCLUDED |
|  | #define HISTOGRAM\_H\_INCLUDED |
|  | #include <vector> |
|  |  |
|  | using namespace std; |
|  |  |
|  | vector<double> input\_numbers(size\_t count); |
|  |  |
|  | vector<size\_t> make\_histogram(vector<double> numbers, size\_t bin\_count); |
|  |  |
|  | void show\_histogram\_text(vector<size\_t> bins); |
|  |  |
|  | bool check\_height (size\_t bin\_height, size\_t bin\_count, size\_t IMAGE\_HEIGHT); |
|  |  |
|  | void find\_minmax(vector<double> numbers, double& min, double& max); |
|  |  |
|  | #endif // HISTOGRAM\_H\_INCLUDED |

***histogram.cpp***

|  |
| --- |
| #include "histogram.h" |
|  | #include <vector> |
|  | #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | vector<double> input\_numbers(size\_t count) |
|  | { |
|  | vector<double> result(count); |
|  | cerr << "Enter numbers: "; |
|  | for (size\_t i = 0; i < count; i++) |
|  | { |
|  | cin >> result[i]; |
|  | } |
|  | return result; |
|  | } |
|  | vector<size\_t> make\_histogram(vector<double> numbers, size\_t bin\_count) |
|  | { |
|  | vector<size\_t> bins(bin\_count); |
|  | double max, min; |
|  | find\_minmax(numbers, min, max); |
|  | for (double number : numbers) |
|  | { |
|  | size\_t bin = (size\_t)((number - min) / (max - min) \* bin\_count); |
|  | if (bin == bin\_count) |
|  | { |
|  | bin--; |
|  | } |
|  | bins[bin]++; |
|  | } |
|  | return bins; |
|  | } |
|  | void show\_histogram\_text(vector<size\_t> bins) |
|  | { |
|  | const size\_t SCREEN\_WIDTH = 80; |
|  | const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1; |
|  | size\_t max\_count = 0; |
|  | for (size\_t count : bins) |
|  | { |
|  | if (count > max\_count) |
|  | { |
|  | max\_count = count; |
|  | } |
|  | } |
|  | const bool scaling\_needed = max\_count > MAX\_ASTERISK; |
|  | for (size\_t bin : bins) |
|  | { |
|  | if (bin < 100) |
|  | { |
|  | cout << ' '; |
|  | } |
|  | if (bin < 10) |
|  | { |
|  | cout << ' '; |
|  | } |
|  | cout << bin << "|"; |
|  | size\_t height = bin; |
|  | if (scaling\_needed) |
|  | { |
|  | const double scaling\_factor = (double)MAX\_ASTERISK / max\_count; |
|  | height = (size\_t)(bin \* scaling\_factor); |
|  | } |
|  | for (size\_t i = 0; i < height; i++) |
|  | { |
|  | cout << '\*'; |
|  | } |
|  | cout << '\n'; |
|  | } |
|  | return; |
|  | } |
|  |  |
|  | bool check\_height (size\_t bin\_height, size\_t bin\_count, size\_t IMAGE\_HEIGHT) |
|  | { |
|  | if (bin\_height\*bin\_count > IMAGE\_HEIGHT) |
|  | return true; |
|  | else |
|  | return false; |
|  | } |
|  |  |
|  | void find\_minmax(vector<double> numbers, double& min, double& max) |
|  | { |
|  | if (numbers.size() == 0){ |
|  | return; |
|  | } |
|  | min = numbers[0]; |
|  | max = numbers[0]; |
|  | for (double number : numbers) |
|  | { |
|  | if (number < min) |
|  | { |
|  | min = number; |
|  | } |
|  | if (number > max) |
|  | { |
|  | max = number; |
|  | } |
|  | } |
|  | return; |
|  | } |

***SVG.h***

|  |
| --- |
| #ifndef SVG\_H\_INCLUDED |
|  | #define SVG\_H\_INCLUDED |
|  | #include <iostream> |
|  | #include <vector> |
|  |  |
|  | using namespace std; |
|  |  |
|  | void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black"); |
|  |  |
|  | void svg\_text(double left, double baseline, string text); |
|  |  |
|  | void svg\_begin(double width, double height); |
|  |  |
|  | void svg\_end(); |
|  |  |
|  | void show\_histogram\_svg(const vector<size\_t>& bins, size\_t bin\_height, size\_t bin\_count); |
|  |  |
|  | #endif // SVG\_H\_INCLUDED |

***SVG.cpp***

|  |
| --- |
| #include "SVG.h" |
|  | #include "histogram.h" |
|  | #include <iostream> |
|  | #include <vector> |
|  |  |
|  | using namespace std; |
|  |  |
|  | void svg\_rect(double x, double y, double width, double height, string stroke, string fill) |
|  | { |
|  | cout << "<rect x='"<< x << "' y='" <<y<<"' width='" <<width <<"' height='" <<height <<"' stroke='"<< stroke <<"' fill='"<<fill<<"'/>\n"; |
|  | } |
|  |  |
|  | void svg\_text(double left, double baseline, string text) |
|  | { |
|  | cout << "<text x='" << left << "' y='"<< baseline <<"'>"<<text<<"</text>\n"; |
|  | } |
|  |  |
|  | void svg\_begin(double width, double height) |
|  | { |
|  | cout << "<?xml version='1.0' encoding='UTF-8'?>\n"; |
|  | cout << "<svg "; |
|  | cout << "width='" << width << "' "; |
|  | cout << "height='" << height << "' "; |
|  | cout << "viewBox='0 0 " << width << " " << height << "' "; |
|  | cout << "xmlns='http://www.w3.org/2000/svg'>\n"; |
|  | } |
|  |  |
|  | void svg\_end() |
|  | { |
|  | cout << "</svg>\n"; |
|  | } |
|  |  |
|  | void show\_histogram\_svg(const vector<size\_t>& bins, size\_t bin\_height, size\_t bin\_count) |
|  | { |
|  | const auto IMAGE\_WIDTH = 400; |
|  | const auto IMAGE\_HEIGHT = 700; |
|  | const auto TEXT\_LEFT = 20; |
|  | const auto TEXT\_BASELINE = 20; |
|  | const auto TEXT\_WIDTH = 50; |
|  | const auto BLOCK\_WIDTH = 10; |
|  | if (! check\_height) |
|  | bin\_height = IMAGE\_HEIGHT / bin\_count; |
|  | svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT); |
|  | double top = 0; |
|  | for (size\_t bin : bins) |
|  | { |
|  | const double bin\_width = BLOCK\_WIDTH \* bin; |
|  | svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin)); |
|  | svg\_rect(TEXT\_WIDTH, top, bin\_width, bin\_height, "blue", "blue"); |
|  | top += bin\_height; |
|  | } |
|  | svg\_end(); |
|  | } |

***test.cpp***

|  |
| --- |
| #include "histogram.h" |
|  | #include <cassert> |
|  |  |
|  | void test\_positive() |
|  | { |
|  | double min = 0; |
|  | double max = 0; |
|  | find\_minmax({1, 2, 3}, min, max); |
|  | assert(min == 1); |
|  | assert(max == 3); |
|  | } |
|  |  |
|  | void |
|  | test\_negative() |
|  | { |
|  | double min = 0; |
|  | double max = 0; |
|  | find\_minmax({-1, -2, -3}, min, max); |
|  | assert(min == -3); |
|  | assert(max == -1); |
|  | } |
|  | void |
|  | test\_same() |
|  | { |
|  | double min = 0; |
|  | double max = 0; |
|  | find\_minmax({3, 3, 3}, min, max); |
|  | assert(min == 3); |
|  | assert(max == 3); |
|  | } |
|  | void |
|  | test\_onenumber() |
|  | { |
|  | double min = 0; |
|  | double max = 0; |
|  | find\_minmax({2}, min, max); |
|  | assert(min == 2); |
|  | assert(max == 2); |
|  | } |
|  | void |
|  | test\_empty() |
|  | { |
|  | double min = 0; |
|  | double max = 0; |
|  | find\_minmax({}, min, max); |
|  | } |
|  |  |
|  | void test\_check\_height\_false () |
|  | { |
|  | size\_t bin\_height = 5; |
|  | size\_t bin\_count = 2; |
|  | size\_t IMAGE\_HEIGHT = 15; |
|  | assert(check\_height(bin\_height, bin\_count, IMAGE\_HEIGHT) == false); |
|  | } |
|  |  |
|  | void test\_check\_height\_true () |
|  | { |
|  | size\_t bin\_height = 5; |
|  | size\_t bin\_count = 5; |
|  | size\_t IMAGE\_HEIGHT = 15; |
|  | assert(check\_height(bin\_height, bin\_count, IMAGE\_HEIGHT) == true); |
|  | } |
|  |  |
|  | int |
|  | main() |
|  | { |
|  | test\_positive(); |
|  | test\_negative(); |
|  | test\_same(); |
|  | test\_onenumber(); |
|  | test\_empty(); |
|  | test\_check\_height\_false(); |
|  | test\_check\_height\_true(); |
|  |  |
|  | } |