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 № 4**

# По курсу «Разработка ПО систем управления»

# «Декомпозиция и контроль корректности программ»

# Выполнил студент группы А-02-19:

# Жданова Е.Д.

# Проверили:

# Мохов А. С.

# Козлюк Д. А.

**1 Цель работы**

1. Уметь устанавливать и подключать к программе внешние библиотеки.
2. Уметь использовать типовые элементы API: функции обратного вызова, битовые флаги и маски, массивы и строки C.
3. Уметь работать с параметрами командной строки программы.

#### Вариант 5

Добавьте программе опцию -generate, при указании которой программа будет генерировать числа вместо их считывания. Количество столбцов считывать по-прежнему нужно. Например, lab03.exe -generate 100 генерирует 100 чисел. Опция может стоять до или после URL: lab03.exe -generate 10 http://... или lab04.exe http://... -generate 10. Если после -generate не стоит числа, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Ссылка на репозитарий:**

**https://github.com/ZhdanovaED/cs-laba03/blob/master/main.cpp**

**Svg.cpp:**

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include "svg.h" |
|  |  |
|  | 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 svg\_text(double left, double baseline, string text) { |
|  | cout << "<text x='"<<left<<"' y='"<<baseline<<"'>"<<text<<"</text>";} |
|  |  |
|  | void svg\_rect(double x, double y, double width, double height,string stroke= "black", string fill= "black"){ |
|  | cout<< "<rect x='"<<x<<"' y='"<<y<<"' width='"<<width<<"' height='"<<height<<"' stroke='"<<stroke<<"' fill='"<<fill<<"'/>";} |
|  |  |
|  | double scaling(const vector<size\_t>& bins, double block, double IMAGE\_WIDTH, double TEXT\_LEFT){ |
|  | double max; |
|  | max=-1; |
|  | for (size\_t bin: bins){ |
|  | if (bin>max) |
|  | max=bin; |
|  | } |
|  |  |
|  | block = ((IMAGE\_WIDTH-TEXT\_LEFT) /max); |
|  | return block; |
|  | } |
|  | size\_t text\_reflection\_check (size\_t IMAGE\_WIDTH , size\_t TEXT\_WIDTH , size\_t TEXT\_LEFT){ |
|  | size\_t text\_reflection; |
|  | return text\_reflection=IMAGE\_WIDTH + TEXT\_WIDTH - TEXT\_LEFT; |
|  | } |
|  | void show\_histogram\_svg(const vector<size\_t>& bins) { |
|  | const auto block= 10; |
|  | const auto IMAGE\_WIDTH = 420; |
|  | const auto TEXT\_LEFT = 40; |
|  | double BLOCK\_WIDTH = scaling(bins, block, IMAGE\_WIDTH, TEXT\_LEFT); |
|  | const auto IMAGE\_HEIGHT = 300; |
|  | const auto TEXT\_BASELINE = 20; |
|  | const auto TEXT\_WIDTH = 10; |
|  | const auto BIN\_HEIGHT = (IMAGE\_HEIGHT/bins.size()); |
|  | size\_t text\_reflection = text\_reflection\_check (IMAGE\_WIDTH, TEXT\_WIDTH , TEXT\_LEFT); |
|  | size\_t histogram\_reflection; |
|  |  |
|  | svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT); |
|  | double top = 0; |
|  | for (size\_t bin : bins) { |
|  | const double bin\_width = BLOCK\_WIDTH \* bin; |
|  | histogram\_reflection = IMAGE\_WIDTH-bin\_width-TEXT\_LEFT; |
|  | svg\_rect(histogram\_reflection , top, bin\_width, BIN\_HEIGHT,"darkviolet","#9400D3"); |
|  | svg\_text(text\_reflection, top + TEXT\_BASELINE, to\_string(bin)); |
|  |  |
|  | top += BIN\_HEIGHT; |
|  | } |
|  | svg\_end(); |
|  | } |

**Svg.h:**

|  |
| --- |
| #ifndef SVG\_H\_INCLUDED |
|  | #define SVG\_H\_INCLUDED |
|  | #include <vector> |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | void svg\_begin(double width, double height); |
|  |  |
|  | void svg\_end(); |
|  |  |
|  | void svg\_text(double left, double baseline, string text); |
|  |  |
|  | void svg\_rect(double x, double y, double width, double height,string stroke, string fill); |
|  |  |
|  |  |
|  | double scaling(const vector<size\_t>& bins, double block, double IMAGE\_WIDTH ); |
|  |  |
|  | void show\_histogram\_svg(const vector<size\_t>& bins); |
|  |  |
|  | size\_t text\_reflection\_check (size\_t IMAGE\_WIDTH , size\_t TEXT\_WIDTH , size\_t TEXT\_LEFT); |
|  |  |
|  |  |
|  | #endif // SVG\_H\_INCLUDED |

**Histogram.cpp:**

|  |
| --- |
| #include "histogram.h" |
|  | #include <vector> |
|  | #include <windows.h> |
|  | void find\_minmax(const vector<double>& numbers, double& min, double& max) |
|  | { |
|  | if(numbers.size()==0) |
|  | { |
|  | return; |
|  | } |
|  | else{ |
|  | min=numbers[0]; |
|  | max=numbers[0]; |
|  | for (size\_t i=1; i<numbers.size(); i++) |
|  | { |
|  | if (min>numbers[i]) |
|  | { |
|  | min=numbers[i]; |
|  | } |
|  | if (max<numbers[i]) |
|  | { |
|  | max=numbers[i]; |
|  | }}} |
|  | } |
|  | string make\_info\_text() { |
|  | stringstream buffer; |
|  | DWORD WINAPI GetVersion(); |
|  | DWORD mask = 0x0000ffff; |
|  | DWORD mask2 = 0x000000ff; |
|  | DWORD info = GetVersion(); |
|  | DWORD platform = info >> 16; |
|  | DWORD version = info & mask; |
|  | DWORD version\_major = version & mask2; |
|  | DWORD version\_minor = version >>8; |
|  | if ((version & 0x40000000) == 0) { |
|  |  |
|  |  |
|  | DWORD build = platform; |
|  |  |
|  | buffer << "Windows v" << version\_major << "." << version\_minor << "(build " << build << ")\n"; |
|  | } |
|  |  |
|  | char computer\_name[MAX\_COMPUTERNAME\_LENGTH + 1]; |
|  | DWORD size = sizeof(computer\_name); |
|  | GetComputerNameA(computer\_name, &size); |
|  | buffer << "Computer name: " << computer\_name << "\n"; |
|  |  |
|  | return buffer.str(); |
|  | } |

**Histogram.h:**

|  |
| --- |
| #ifndef HISTOGRAM\_H\_INCLUDED |
|  | #define HISTOGRAM\_H\_INCLUDED |
|  | #include <vector> |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <sstream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | void find\_minmax(const vector<double>& numbers, double& min, double& max); |
|  |  |
|  | string make\_info\_text(); |
|  |  |
|  | vector<size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count); |
|  |  |
|  | #endif // HISTOGRAM\_H\_INCLUDED |

**Main.cpp:**

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include "histogram.h" |
|  | #include "svg.h" |
|  | #include <cstdio> |
|  | #include <windows.h> |
|  | #include <string> |
|  | #include <sstream> |
|  | using namespace std; |
|  |  |
|  | vector<double> input\_numbers(size\_t count) { |
|  | vector<double> result(count); |
|  | for (size\_t i = 0; i < count; i++) { |
|  | cin >> result[i]; |
|  | } |
|  | return result; |
|  | } |
|  |  |
|  | vector<size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count){ |
|  | double min, max; |
|  | find\_minmax(numbers, min, max); |
|  | vector<size\_t>count(bin\_count,0); |
|  | for (double x: numbers){ |
|  | size\_t bin\_index=(size\_t)(x-min)\*(bin\_count)/(max-min); |
|  | if (bin\_index==bin\_count) |
|  | { |
|  | bin\_index--; |
|  | } |
|  | count[bin\_index]++;} |
|  | return count; |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | DWORD WINAPI GetVersion(); |
|  | DWORD mask = 0x0000ffff; |
|  | DWORD mask2 = 0x000000ff; |
|  | DWORD info = GetVersion(); |
|  | DWORD platform = info >> 16; |
|  | DWORD version = info & mask; |
|  | DWORD version\_major = version & mask2; |
|  | DWORD version\_minor = version >>8; |
|  | if ((version & 0x80000000) == 0) { |
|  | printf("true\n"); |
|  | } |
|  |  |
|  | DWORD build = platform; |
|  |  |
|  | printf("Windows v%d.%d (build %d)", version\_major, version\_minor, build); |
|  |  |
|  | return 0; |
|  |  |
|  | size\_t number\_count; |
|  | cerr<< "enter number count"; |
|  | cin>>number\_count; |
|  | const auto numbers=input\_numbers(number\_count); |
|  |  |
|  | size\_t bin\_count; |
|  | cerr<< "enter bin count"; |
|  | cin>>bin\_count; |
|  | const auto bins = make\_histogram(numbers, bin\_count); |
|  | show\_histogram\_svg(bins); |
|  | return 0; |
|  |  |
|  | } |