Федеральное государственное бюджетное образовательное учреждение высшего образования   
"Национальный исследовательский университет "МЭИ"

РАЗРАБОТКА ПРОГРАММНОГО ОБСЕПЕЧЕНИЯ СИСТЕМ УПРАВЛЕНИЯ

ЛАБОРАТОРНАЯ РАБОТА №4  
«Библиотеки, низкоуровневые операции»

Студент:  
Никифорова Екатерина Владимировна  
Группа А-03-19

Преподаватель:   
Мохов Андрей Сергеевич

Москва 2020

Вариант 13

Постановка индивидуального задания

Добавьте программе опцию -stroke для задания цвета границ столбцов. Например, lab03.exe -stroke red или lab04.exe -stroke "#ff0000" делает их красными. Опция может стоять до или после URL: lab03.exe -stroke red http://... или lab04.exe http://... -fill red. Если после -stroke не стоит еще одного аргумента, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

Логика решения

Для правильного использования в программе опции -stroke нужно добавить структуру, в которой будет храниться цвет и нарушения, если есть. Кроме того, нужна новая функция, которая будет отвечать за считывание и проверку. Требуется также добавить в определение функции show\_histogram\_svg() передачу значения переменной stroke и добавить новый параметр в Input download().

1) Код модуля main.cpp

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg\_histogram.h"

#include <curl/curl.h>

#include <windows.h>

#include <string>

#include <sstream>

using namespace std;

struct Options

{

string stroke;

bool stroke\_veracity;

bool violation;

char\* url;

};

Options parse\_args(int argc, char\*\* argv)

{

Options opt;

opt.url=0;

opt.stroke\_veracity = false;

opt.violation = false;

for (int i = 1; i < argc; i++)

{

if (argv[i][0] == '-')

{

if(string(argv[i]) == "-stroke")

{

if(i + 1 < argc)

{

opt.stroke = string(argv[i+1]);

if (opt.stroke.size())

{

opt.stroke\_veracity = true;

i++;

}

else

{

opt.violation = true;

}

}

else

{

opt.violation = true;

}

}

}

else

{

opt.url = argv[i];

}

}

return opt;

}

vector<double> input\_numbers(istream& in, const size\_t count)

{

vector<double> result(count);

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

{

in >> result[i];

}

return result;

}

Input read\_input(istream& in, bool prompt)

{

Input data;

size\_t number\_count;

if (prompt)

{

cerr << "Enter number count: ";

in >> number\_count;

cerr << "Enter numbers: ";

data.numbers = input\_numbers(in, number\_count);

cerr << "Enter column count: ";

in >> data.bin\_count;

}

else

{

in >> number\_count;

data.numbers = input\_numbers(in, number\_count);

in >> data.bin\_count;

}

return data;

}

size\_t write\_data(void\* items, size\_t item\_size, size\_t item\_count, void\* ctx)

{

const size\_t data\_size = item\_size \* item\_count;

const char\* new\_items = reinterpret\_cast<const char\*>(items);

stringstream\* buffer = reinterpret\_cast<stringstream\*>(ctx);

buffer->write(new\_items, data\_size);

return data\_size;

}

Input download(const string& address, const Options &opt)

{

stringstream buffer;

curl\_global\_init(CURL\_GLOBAL\_ALL);

CURL\* curl = curl\_easy\_init();

if(curl)

{

CURLcode res;

curl\_easy\_setopt(curl, CURLOPT\_URL, address.c\_str());

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_data);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, &buffer);

res = curl\_easy\_perform(curl);

if (res != CURLE\_OK)

{

cout << curl\_easy\_strerror(res) << endl;

exit(1);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

/\*void show\_histogram\_text(vector<size\_t> bins, size\_t number\_count)

{

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';

}

}\*/

int main(int argc, char\* argv[])

{

Input input;

Options opt = parse\_args(argc,argv);

if (opt.violation)

{

cerr<<"Error";

return 1;

}

if (opt.url)

{

input = download(opt.url,opt);

}

else

{

input = read\_input(cin, true);

}

const auto bins = make\_histogram(input);

show\_histogram\_svg(bins,opt.stroke);

return 0;

}

2) Код модуля histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

};

void find\_minmax(const vector<double>& numbers, double& min, double& max);

vector<size\_t> make\_histogram(const Input& data);

#endif // HISTOGRAM\_H\_INCLUDED

3) Код модуля histogram.cpp

#include "histogram.h"

#include <iostream>

#include <vector>

using namespace std;

void find\_minmax(const vector<double>& numbers, double& min, double& max)

{

if (numbers.size()==0)

{

return;

}

else

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

vector<size\_t> make\_histogram(const Input& data)

{

vector<size\_t> result(data.bin\_count);

double min;

double max;

find\_minmax(data.numbers, min, max);

for (double number : data.numbers)

{

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

if (bin == data.bin\_count)

{

bin--;

}

result[bin]++;

}

return result;

}

4) Код модуля svg\_histogram.h

#ifndef SVG\_HISTOGRAM\_H\_INCLUDED

#define SVG\_HISTOGRAM\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

void svg\_begin(double width, double height);

void svg\_end();

void svg\_rect(double x, double y, double width, double height,string stroke,string fill );

void svg\_text(double left, double baseline, string text);

void poisk\_max\_count(const vector<size\_t>& bins, double& max\_count);

void show\_histogram\_svg(const vector<size\_t>& bins, string stroke);

#endif // SVG\_HISTOGRAM\_H\_INCLUDED

5) Код модуля svg\_histogram.cpp

#include <iostream>

#include <vector>

#include "svg\_histogram.h"

#include <windows.h>

#include <sstream>

#include <string>

using namespace std;

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\_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 << "' />";

}

void svg\_text(double left, double baseline, string text)

{

cout << "<text x='" << left << "' y='" << baseline << "'>" << text << "</text>";

}

void poisk\_max\_count(const vector<size\_t>& bins, double& max\_count)

{

if (bins.size() == 0)

{

return;

}

else

{

max\_count = bins[0];

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

}

}

string make\_info\_text()

{

stringstream buffer;

DWORD mask = 0x0000ffff;

DWORD mask\_major = 0b00000000'00000000'00000000'11111111;

DWORD info = GetVersion();

DWORD build;

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & mask\_major;

DWORD version\_minor = version >> 8;

if ((info & 0x80000000) == 0)

{

build = platform;

buffer << "Windows v" << version\_major << "." << version\_minor << "(build " << build << ")\n";

}

char system\_name[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD size = sizeof(system\_name);

GetComputerNameA(system\_name, &size);

//printf("Computer name: %s\n", system\_name);

buffer << "Computer name: " << system\_name;

return buffer.str();

}

void show\_histogram\_svg(const vector<size\_t>& bins, string stroke)

{

const auto IMAGE\_WIDTH = 600;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_HEIGHT = 40;

const auto BLOCK\_WIDTH = 30;

const auto TEXT\_WIDTH = BLOCK\_WIDTH / 2;

svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);

double top = 0;

string fill = "green";

double max\_count;

poisk\_max\_count(bins, max\_count);

double max\_height = 0;

double bottom = 0;

max\_height = IMAGE\_HEIGHT - 2 \* TEXT\_HEIGHT;

const double scaling\_factor = (double)(IMAGE\_HEIGHT - 2 \* TEXT\_HEIGHT) / max\_count;

for (size\_t bin : bins)

{

const double bin\_height = bin \* scaling\_factor;

top = max\_height - bin\_height;

svg\_rect(bottom, top, BLOCK\_WIDTH, bin\_height, stroke, "green");

svg\_text(bottom + TEXT\_WIDTH, max\_height + TEXT\_HEIGHT, to\_string(bin));

bottom += BLOCK\_WIDTH;

}

bottom = 0;

string info = make\_info\_text();

svg\_text(bottom + TEXT\_WIDTH, max\_height + 2 \* TEXT\_HEIGHT, info);

svg\_end();

}

Ссылка на репозитарий

https://github.com/NikiforovaEV/cs-lab03