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»

# «Библиотеки и низкоуровневые операции»

# *Выполнил:*

# Студент группы А-01-19

# Маркелов Н.А.

# *Проверили:*

# Мохов А. С

# Козлюк Д. А

**Москва 2020**

**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.
4. Уметь применять побитовые операции для типовых сценариев.
5. Уметь работать с API, принимающими указатели, в том числе строки C.
6. Знать характерные особенности документации на API библиотек.

# Задание:

# Часть 1. Библиотеки

Добавить возможность построения гистограммы по данным из файла из сети. Адрес файла задается аргументом командной строки программы. Если адрес не задан, читать данные со стандартного ввода, как раньше.

Работу нужно вести на основе кода общего задания к ЛР № 3 в старом репозитарии в отдельной ветке без создания нового проекта. По этой причине во всех примерах используется lab03.exe.

# Часть 2. Низкоуровневые операции

# Код, который будет написан в этой части работы, должен в итоге формировать строку, которую нужно отобразить снизу итогового SVG в формате:

Windows v5.1 (build 1234)

Computer name: My-Comp

**Индивидуальное задание**

**Вариант 10**

Добавьте программе опцию -verbose, при указании которой нужно включать отладочный вывод cURL с помощью CURLOPT\_VERBOSE. Опция может стоять до или после URL: lab03.exe -verbose http://... или lab04.exe http://... -verbose. Если встретился иной аргумент, начинающийся с дефиса, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Логика решения варианта:**

Для осуществления поставленной задачи, программа должна анализировать переданные в неё аргументы и в зависимости от них осуществлять отладочный вывод, вывод подсказки и т.д.

Была создана структура, состоящая из 3х полей, первые 2 отвечают за необходимость отладочного вывода и вывода подсказки для пользователя, 3е хранит в себе возможный URL страницы (если он будет передан, в качестве аргумента), с которой будет производиться загрузка данных, для построения гистограммы.

После инициализации вышеописанной структуры в теле программы, она заполняется с помощью специальной функции, в которую передаются аргументы программы. Логика работы функции: 1) изначально всем логическим полям присваивается «false» 2) Затем в цикле идёт перебор переданных аргументов, в случае соответствия, соответствующим полям структуры присваивается «true» 3) Для определения «неизвестной» опции (после которой должна выводиться подсказка) использовалась функция сравнения 2х строк strcmp (первый символ должен быть «-», последующие символы не должны являться строкой «verbose»).

В теле основной программы тоже были сделаны изменения. Теперь, если поле, отвечающее за необходимость подсказки равно «true», то программа выводит подсказку и завершается досрочно. Если в параметры был передан URL, то программа вызывает видоизменённую версию функции download (в параметры добавлен bool UseVerbose), иначе ввод идет с клавиатуры.

**Исходный код всех модулей**

**main.cpp**

**===================================================================**

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include "histogram.h" |
|  | #include "svg.h" |
|  | #include <curl/curl.h> |
|  | #include <cstdio> |
|  | #include <windows.h> |
|  |  |
|  | using namespace std; |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | Input input; |
|  | Configurations config=input\_config(argc, argv); |
|  |  |
|  | //Hint for user |
|  | if (config.UseHint) |
|  | { |
|  | cerr << "\n###########################################################"<< |
|  | "\nHELP:\n \*For making input from web enter URL of page to parameters"<< |
|  | "\n \*For using CURLOPT\_VERBOSE mode enter ''-verbose''" << "\n" |
|  | << "###########################################################\n"; |
|  | exit(2); |
|  | } |
|  |  |
|  | //input data |
|  | if (config.UrlPage) |
|  | { |
|  | input=download(config.UrlPage, config.UseVerbose); |
|  | } |
|  | else |
|  | { |
|  | input=read\_input(cin, true); |
|  | } |
|  |  |
|  | //making histogram |
|  | const auto bins=make\_histogram(input); |
|  | double bin\_height; //For module tests |
|  | show\_histogram\_svg(bins, bin\_height); |
|  |  |
|  | return 0; |
|  | } |

**===================================================================**

**histogram.cpp**

|  |
| --- |
| #include "histogram.h" |
|  |  |
|  | Input read\_input(istream& in, bool prompt) |
|  | { |
|  | Input data; |
|  |  |
|  | if (prompt) |
|  | { |
|  | cerr << "Enter number count: "; |
|  | } |
|  | size\_t number\_count; |
|  | in >> number\_count; |
|  |  |
|  | if (prompt) |
|  | { |
|  | cerr << "Enter numbers: "; |
|  | } |
|  | data.numbers = input\_numbers(in, number\_count); |
|  |  |
|  | if (prompt) |
|  | { |
|  | cerr << "Enter column count: "; |
|  | } |
|  | in >> data.bin\_count; |
|  |  |
|  | return data; |
|  | } |
|  |  |
|  | size\_t write\_data(void\* items, size\_t item\_size, size\_t item\_count, void\* ctx) |
|  | { |
|  | size\_t data\_size = item\_size \* item\_count; |
|  | stringstream\* buffer = reinterpret\_cast<stringstream\*>(ctx); |
|  | (\*buffer).write(reinterpret\_cast<const char\*>(items), data\_size); |
|  | return data\_size; |
|  | } |
|  |  |
|  | Input download(const string& address, bool UseVerbose) |
|  | { |
|  | stringstream buffer; |
|  |  |
|  | curl\_global\_init(CURL\_GLOBAL\_ALL); |
|  | CURL\* curl = curl\_easy\_init(); |
|  | if(curl) |
|  | { |
|  | CURLcode res; |
|  | if (UseVerbose) |
|  | { |
|  | curl\_easy\_setopt(curl, CURLOPT\_VERBOSE, 1L); |
|  | } |
|  | 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) |
|  | { |
|  | cerr << curl\_easy\_strerror(res) << endl; |
|  | exit(1); |
|  | } |
|  | } |
|  | curl\_easy\_cleanup(curl); |
|  | return read\_input(buffer, false); |
|  |  |
|  | } |
|  |  |
|  | vector<double> input\_numbers(istream& in, size\_t count) |
|  | { |
|  | vector<double> result(count); |
|  | for (size\_t i = 0; i < count; i++) |
|  | { |
|  | in >> result[i]; |
|  | } |
|  | return result; |
|  | } |
|  |  |
|  | void find\_minmax(vector<double> numbers, double& min, double& max) |
|  | { |
|  | min = numbers[0]; |
|  | max = numbers[0]; |
|  | for (auto number : numbers) |
|  | { |
|  | if (number < min) |
|  | { |
|  | min = number; |
|  | } |
|  | if (number > max) |
|  | { |
|  | max = number; |
|  | } |
|  | } |
|  | return; |
|  | } |
|  |  |
|  | vector<size\_t> make\_histogram(const Input input) |
|  | { |
|  | double min, max; |
|  | find\_minmax(input.numbers, min, max); |
|  |  |
|  | vector<size\_t> bins(input.bin\_count); |
|  | for (double number : input.numbers) |
|  | { |
|  | size\_t bin = (size\_t)((number - min) / (max - min) \* input.bin\_count); |
|  | if (bin == input.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; |
|  | } |
|  |  |
|  | Configurations input\_config(int argc, char\*\* argv) |
|  | { |
|  | Configurations config; |
|  |  |
|  | config.UseVerbose=false; |
|  | config.UseHint=false; |
|  | config.UrlPage=0; |
|  |  |
|  | for (int i=1; i<argc; i++) |
|  | { |
|  | if (argv[i][0]=='-') |
|  | { |
|  | if (strcmp(argv[i], "-verbose") == 0) |
|  | { |
|  | config.UseVerbose=true; |
|  | } |
|  | else |
|  | { |
|  | config.UseHint=true; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | config.UrlPage=argv[i]; |
|  | } |
|  | } |
|  |  |
|  | return config; |
|  | } |

**===================================================================**

**histogram.h**

|  |
| --- |
| #pragma once |
|  |  |
|  | #include <iostream> |
|  | #include <vector> |
|  | #include <sstream> |
|  | #include <string> |
|  | #include <curl/curl.h> |
|  | using namespace std; |
|  |  |
|  | struct Input |
|  | { |
|  | vector<double> numbers; |
|  | size\_t bin\_count; |
|  | }; |
|  |  |
|  | struct Configurations |
|  | { |
|  | bool UseVerbose; |
|  | bool UseHint; |
|  | char\* UrlPage; |
|  | }; |
|  |  |
|  | Input read\_input(istream& in, bool prompt); |
|  | size\_t write\_data(void\* items, size\_t item\_size, size\_t item\_count, void\* ctx); |
|  | Input download(const string& address, bool UseVerbose); |
|  | vector<double> input\_numbers(istream& in, size\_t count); |
|  | void find\_minmax(vector<double> numbers, double& min, double& max); |
|  | vector<size\_t> make\_histogram(const Input input); |
|  | void show\_histogram\_text(vector<size\_t> bins); |
|  | Configurations input\_config(int argc, char\*\* argv); |

**===================================================================**

**svg.cpp**

|  |
| --- |
| #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, string fil) |
|  | { |
|  | cout << "<rect x='" << x <<"' y='" << y << "' width='" << width << "' height='" << height |
|  | << "' stroke='" << stroke << "' fill='" << fil << "' />"; |
|  | } |
|  |  |
|  | void show\_histogram\_svg(const vector<size\_t>& bins, double& bin\_height) |
|  | { |
|  |  |
|  | const auto IMAGE\_WIDTH = 500; |
|  | const auto IMAGE\_HEIGHT = 300; |
|  | const auto TEXT\_TOP = 20; |
|  | const auto TEXT\_HEIGHT = 30; |
|  | const auto BIN\_WIDTH = 30; |
|  | const auto TEXT\_BASELINE = BIN\_WIDTH / 2; |
|  |  |
|  | svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT); |
|  |  |
|  | double left =0; |
|  |  |
|  | size\_t max\_count=bins[0]; |
|  |  |
|  | for (size\_t bin : bins) |
|  | { |
|  | if (bin>max\_count) |
|  | { |
|  | max\_count=bin; |
|  | } |
|  | } |
|  |  |
|  | for (size\_t bin: bins) |
|  | { |
|  | const double bin\_koeff = (double)(IMAGE\_HEIGHT - TEXT\_HEIGHT) / max\_count; |
|  | bin\_height = bin \* bin\_koeff; |
|  |  |
|  | svg\_text(left + TEXT\_BASELINE, TEXT\_TOP+TEXT\_TOP, to\_string(bin)); |
|  | svg\_rect(left, TEXT\_HEIGHT + TEXT\_TOP, BIN\_WIDTH, bin\_height, "red", "#ffeeee"); |
|  | left += BIN\_WIDTH; |
|  | } |
|  | svg\_text(0, TEXT\_TOP, make\_info\_text()); |
|  | svg\_end(); |
|  | } |
|  |  |
|  | string make\_info\_text() |
|  | { |
|  | stringstream buffer; |
|  |  |
|  | DWORD info = GetVersion(); |
|  | DWORD mask = 0x0000ffff; |
|  | DWORD version = info & mask; |
|  | DWORD platform = info >> 16; |
|  | DWORD mask\_major = 0x0000ff; |
|  |  |
|  | if ((info & 0x80000000) == 0) |
|  | { |
|  | DWORD version\_major = version & mask\_major; |
|  | DWORD version\_minor = version >> 8; |
|  | DWORD build = platform; |
|  |  |
|  | buffer << "Windows v" << version\_major << "." << version\_minor << " (build " << build << ") | "; |
|  | } |
|  |  |
|  | char system\_dir[MAX\_PATH]; |
|  | char computer\_name[MAX\_COMPUTERNAME\_LENGTH + 1]; |
|  | DWORD size = MAX\_COMPUTERNAME\_LENGTH+1; |
|  | GetComputerNameA(computer\_name, &size); |
|  |  |
|  | buffer << "Computer name: " << computer\_name; |
|  |  |
|  | return buffer.str(); |
|  | } |

**===================================================================**

**svg.h**

|  |
| --- |
| #pragma once |
|  |  |
|  | #include <iostream> |
|  | #include <vector> |
|  | #include <sstream> |
|  | #include <windows.h> |
|  |  |
|  | 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 = "black", string fil = "black"); |
|  | void show\_histogram\_svg(const vector<size\_t>& bins, double& bin\_height); |
|  | string make\_info\_text(); |

**===================================================================**

**test.cpp**

|  |
| --- |
| #include "C:\Users\markn\OneDrive\Desktop\Lab3\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; |
|  | double max; |
|  | find\_minmax({-1, -5,-228}, min, max); |
|  | assert(min == -228); |
|  | assert(max == -1); |
|  | } |
|  |  |
|  | void test\_one\_number() |
|  | { |
|  | double min; |
|  | double max; |
|  | find\_minmax({666}, min, max); |
|  | assert(min == 666); |
|  | assert(max == 666); |
|  | } |
|  |  |
|  | void test\_same\_numbers() |
|  | { |
|  | double min; |
|  | double max; |
|  | find\_minmax({3,3,3}, min, max); |
|  | assert(min == 3); |
|  | assert(max == 3); |
|  | } |
|  |  |
|  | void test\_empty\_vector() |
|  | { |
|  | double min; |
|  | double max; |
|  | find\_minmax({}, min, max); |
|  | assert(min == 0); |
|  | assert(max == 0); |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | test\_positive(); |
|  | test\_negative(); |
|  | test\_one\_number(); |
|  | test\_same\_numbers(); |
|  | test\_empty\_vector(); |
|  | } |

**===================================================================**

**test2.cpp**

|  |
| --- |
| #include "C:\Users\markn\OneDrive\Desktop\Lab3\svg.h" |
|  |  |
|  | #include <cassert> |
|  |  |
|  | void scalling\_same() |
|  | { |
|  |  |
|  | double height\_1; |
|  | show\_histogram\_svg({1,1,1},height\_1); |
|  | double height\_2; |
|  | show\_histogram\_svg({322,322,322}, height\_2); |
|  |  |
|  | assert(height\_1==height\_2); |
|  |  |
|  | } |
|  |  |
|  | void scalling\_correct() |
|  | { |
|  |  |
|  | double height\_1; |
|  | show\_histogram\_svg({1, 2, 3}, height\_1); |
|  |  |
|  | double height\_2; |
|  | show\_histogram\_svg({3, 2, 1}, height\_2); |
|  |  |
|  | assert(height\_1>height\_2); |
|  |  |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | scalling\_same(); |
|  | scalling\_correct(); |
|  | } |

**Ссылка на репозитарий**: <https://github.com/marknik139/Lab3>