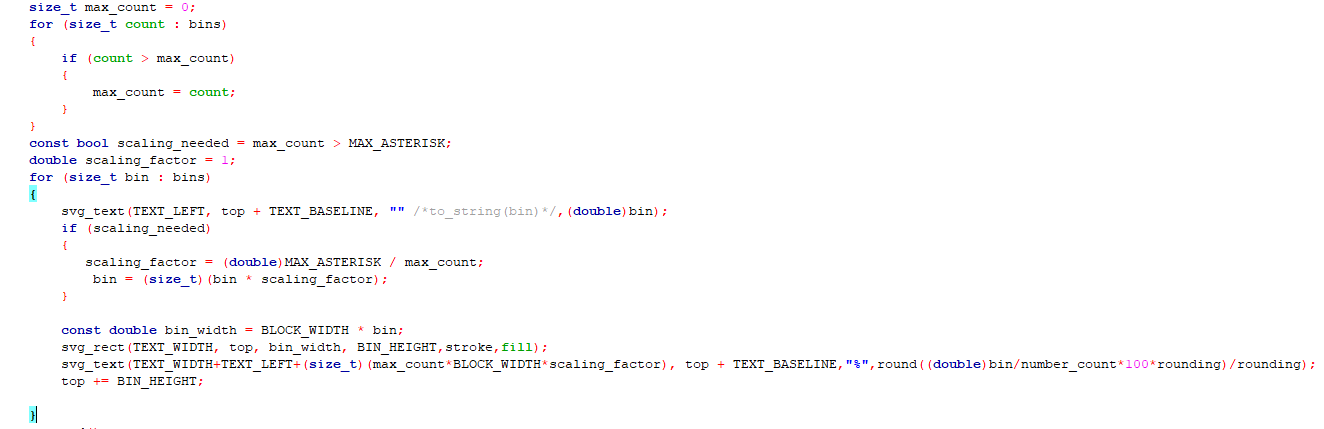
Отчёт по лабораторной работе №3 по теме «Декомпозиция программ»

по предмету

Разработка Программного Обеспечения Систем Управления

Изотова Артёма Александровича, А-03-19

1. Реализацию масштабирования для отрисовки прямоугольников сделал аналогично масштабированию для звёздочек, для начала нахожу максимальное количество чисел в корзине и узнаю нужно ли масштабирование и уже в цикле отрисовки проверяю если масштабирование необходимо, то bin умножаю на коэффициент масштабирования, который отвечает за ширину прямоугольников:



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;

double scaling\_factor = 1;

for (size\_t bin : bins)

{

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, "" ,(double)bin);

if (scaling\_needed)

{

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

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

}

const double bin\_width = BLOCK\_WIDTH \* bin;

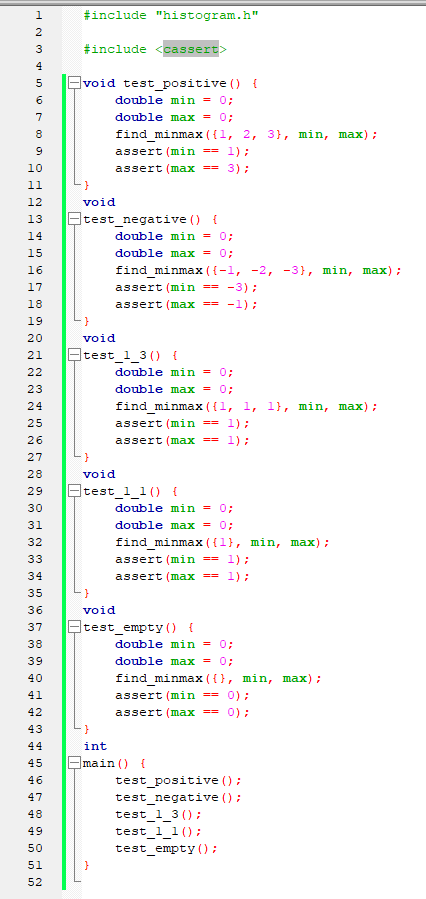
svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill);

svg\_text(TEXT\_WIDTH+TEXT\_LEFT+(size\_t)(max\_count\*BLOCK\_WIDTH\*scaling\_factor), top + TEXT\_BASELINE,"%",round((double)bin/number\_count\*100\*rounding)/rounding);

top += BIN\_HEIGHT;

}

1. Реализация тестов. Для реализации мы создали test.cpp и подключили к нему histogram.h, с помощью которого мы сможем использовать функцию find\_minmax для проверки. Также мы подключили cassert это модуль для проверки ошибок, в частности мы используем функцию assert, которая сравнивает ожидаемые значения с реальными и в случае, если это не так программа выдаёт ошибку, и мы прогоним 5 тестов, где у нас массив положительных чисел, отрицательных, массив из одинаковых чисел, из одного числа , и без чисел.



#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\_1\_3() {

double min = 0;

double max = 0;

find\_minmax({1, 1, 1}, min, max);

assert(min == 1);

assert(max == 1);

}

void

test\_1\_1() {

double min = 0;

double max = 0;

find\_minmax({1}, min, max);

assert(min == 1);

assert(max == 1);

}

void

test\_empty() {

double min = 0;

double max = 0;

find\_minmax({}, min, max);

assert(min == 0);

assert(max == 0);

}

int

main() {

test\_positive();

test\_negative();

test\_1\_3();

test\_1\_1();

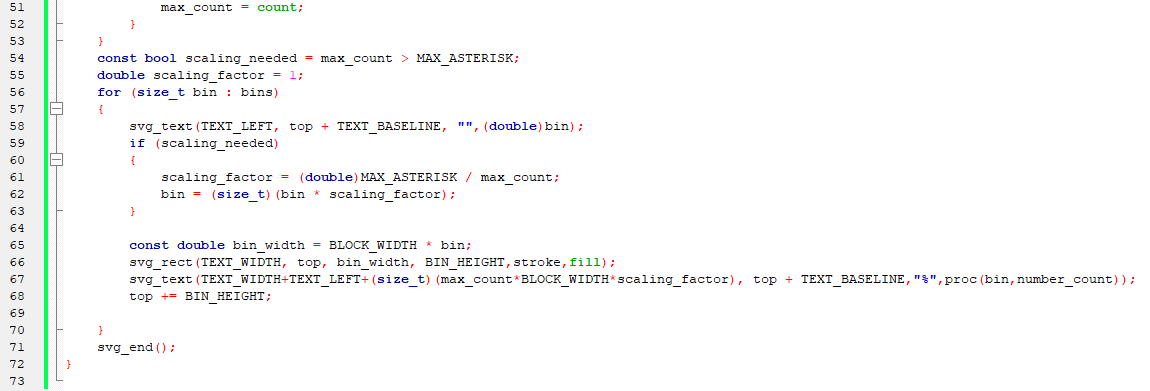
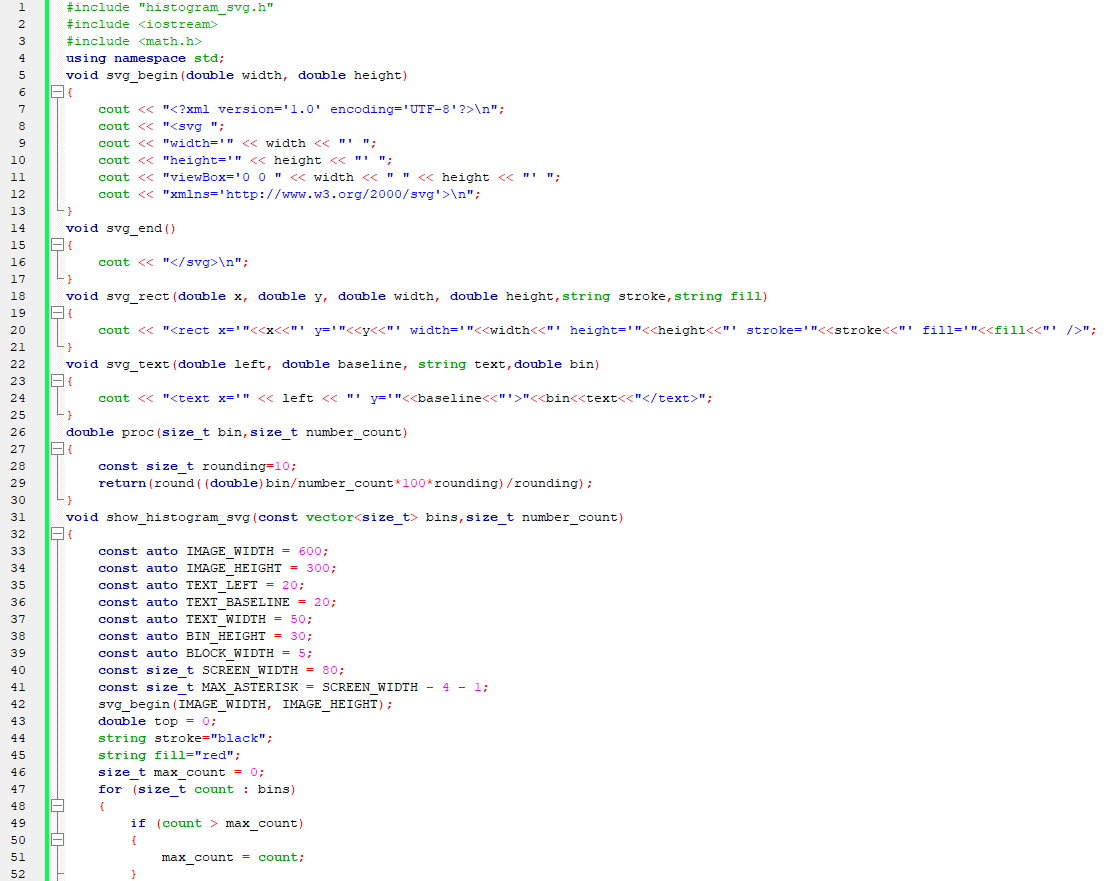
test\_empty();

}

1. Реализация моего варианта дз. Мне было необходимо добавить после прямоугольников проценты чисел входящих в корзину от общего количества чисел, причём они должны были быть на одном уровне и округлены до одного числа после запятой. Для этого мы рассчитываем отступ от левого края причём, рассчитывать необходимо всегда от максимального прямоугольника, чтоб они были на одном уровне и изменятся будет только высота по y, причём также необходимо учесть масштабирование, если максимальный столбец уменьшен, то и расстояние тоже должно уменьшится.

TEXT\_WIDTH+TEXT\_LEFT+(size\_t)(max\_count\*BLOCK\_WIDTH\*scaling\_factor)

Вот такой параметр я передаю для нахождение отступа по Х , приведение типов необходимо так как расстояние должно быть целым. Для реализации округления я подключил модуль math.h, так как в нём есть функция round ,которая округляет в сторону правильного математического округления.



#include "histogram\_svg.h"

#include <iostream>

#include <math.h>

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,double bin)

{

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

}

double proc(size\_t bin,size\_t number\_count)

{

const size\_t rounding=10;

return(round((double)bin/number\_count\*100\*rounding)/rounding);

}

void show\_histogram\_svg(const vector<size\_t> bins,size\_t number\_count)

{

const auto IMAGE\_WIDTH = 600;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 5;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

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

double top = 0;

string stroke="black";

string fill="red";

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;

double scaling\_factor = 1;

for (size\_t bin : bins)

{

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, "",(double)bin);

if (scaling\_needed)

{

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

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

}

const double bin\_width = BLOCK\_WIDTH \* bin;

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill);

svg\_text(TEXT\_WIDTH+TEXT\_LEFT+(size\_t)(max\_count\*BLOCK\_WIDTH\*scaling\_factor), top + TEXT\_BASELINE,"%",proc(bin,number\_count));

top += BIN\_HEIGHT;

}

svg\_end();

}

Код всех модулей

Histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

Main.cpp

#include "histogram.h"

#include "histogram\_svg.h"

#include <iostream>

#include <vector>

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(vector <double> numbers,size\_t bin\_count,double min,double max)

{

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

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,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<<' ';

for (size\_t i=height;i<max\_count+1;i++)

{

cout<<" ";

}

cout<<(double)bin/number\_count\*100<<"%";

cout << '\n';

}

}

int main()

{

// Ввод данных

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

cerr << "Enter numbers: ";

const auto numbers=input\_numbers(number\_count);

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

// Обработка данных

double min;

double max;

find\_minmax(numbers,min,max);

const auto bins = make\_histogram(numbers, bin\_count,min,max);

show\_histogram\_svg(bins,number\_count);

// Вывод данных

return 0;

}

Histogram.cpp

**#include "histogram.h"**

**#include "histogram\_svg.h"**

**#include <iostream>**

**#include <vector>**

**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(vector <double> numbers,size\_t bin\_count,double min,double max)**

**{**

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

**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,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<<' ';**

**for (size\_t i=height;i<max\_count+1;i++)**

**{**

**cout<<" ";**

**}**

**cout<<(double)bin/number\_count\*100<<"%";**

**cout << '\n';**

**}**

**}**

**int main()**

**{**

**// Ввод данных**

**size\_t number\_count;**

**cerr << "Enter number count: ";**

**cin >> number\_count;**

**cerr << "Enter numbers: ";**

**const auto numbers=input\_numbers(number\_count);**

**size\_t bin\_count;**

**cerr << "Enter column count: ";**

**cin >> bin\_count;**

**// Обработка данных**

**double min;**

**double max;**

**find\_minmax(numbers,min,max);**

**const auto bins = make\_histogram(numbers, bin\_count,min,max);**

**show\_histogram\_svg(bins,number\_count);**

**// Вывод данных**

**return 0;**

**}**

**Histogram.cpp**

**#include "histogram.h"**

**#include <iostream>**

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

**{**

**if(numbers.size() !=0)**

**{**

**min = numbers[0];**

**max = numbers[0];**

**for (double number : numbers)**

**{**

**if (number < min)**

**{**

**min = number;**

**}**

**if (number > max)**

**{**

**max = number;**

**}**

**}**

**}**

**// (здесь код поиска минимума и максимума)**

**}**

**Histogram\_svg.cpp**

#include "histogram\_svg.h"

#include <iostream>

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,double bin)

{

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

}

/\*double proc(size\_t bin,size\_t number\_count)

{

const size\_t rounding=10;

return(round((double)bin/number\_count\*100\*rounding)/rounding);

}\*/

void show\_histogram\_svg(const vector<size\_t> bins,size\_t number\_count)

{

const auto IMAGE\_WIDTH = 600;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 5;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

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

double top = 0;

string stroke="black";

string fill="red";

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;

double scaling\_factor = 1;

for (size\_t bin : bins)

{

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, "",(double)bin);

if (scaling\_needed)

{

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

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

}

const double bin\_width = BLOCK\_WIDTH \* bin;

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill);

svg\_text(TEXT\_WIDTH+TEXT\_LEFT+(size\_t)(max\_count\*BLOCK\_WIDTH\*scaling\_factor), top + TEXT\_BASELINE,"%",proc(bin,number\_count));

top += BIN\_HEIGHT;

}

svg\_end();

}

**Histogram\_svg.h**

**#ifndef HISTOGRAM\_SVG\_H\_INCLUDED**

**#define HISTOGRAM\_SVG\_H\_INCLUDED**

**#include <vector>**

**#include <iostream>**

**#include <math.h>**

**using namespace std;**

**void show\_histogram\_svg(const vector<size\_t> bins,size\_t number\_count);**

**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,size\_t bin);**

**double proc(size\_t bin,size\_t number\_count);**

**#endif // HISTOGRAM\_SVG\_H\_INCLUDED**

**Test.cpp**

**#include "histogram.h"**

**#include "histogram\_svg.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\_1\_3()**

**{**

**double min = 0;**

**double max = 0;**

**find\_minmax({1, 1, 1}, min, max);**

**assert(min == 1);**

**assert(max == 1);**

**}**

**void**

**test\_1\_1()**

**{**

**double min = 0;**

**double max = 0;**

**find\_minmax({1}, min, max);**

**assert(min == 1);**

**assert(max == 1);**

**}**

**void**

**test\_empty()**

**{**

**double min = 0;**

**double max = 0;**

**find\_minmax({}, min, max);**

**assert(min == 0);**

**assert(max == 0);**

**}**

**void**

**test\_proc()**

**{**

**size\_t bin=10;**

**size\_t number\_count=100;**

**double p = proc(bin,number\_count);**

**assert(p == 10);**

**}**

**test\_proc2()**

**{**

**size\_t bin=100;**

**size\_t number\_count=0;**

**double p = proc(bin,number\_count);**

**assert(p == 0);**

**}**

**int**

**main()**

**{**

**test\_positive();**

**test\_negative();**

**test\_1\_3();**

**test\_1\_1();**

**test\_empty();**

**test\_proc();**

**test\_proc();**

**}**

**Procent.cpp**

**#include "histogram\_svg.h"**

**#include <math.h>**

**double proc(size\_t bin,size\_t number\_count)**

**{**

**const size\_t rounding=10;**

**if((bin==0)||(number\_count==0))**

**{**

**return(0);**

**}**

**else**

**{**

**return(round((double)bin/number\_count\*100\*rounding)/rounding);**

**}**

**}**