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-02-20

# Сазонова В.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2021

**Задание.**

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

2) Доработать программу в соответствии с вариантом.

#### Вариант 18

Позволять пользователю делать оформление текста - подчеркивание, надчеркивание, ~~зачеркивание~~ текста. За оформление шрифта отвечает атрибут text-decoration. Допустимые значения: none, underline, overline, line-through. Проверять введенной пользователем значение, и если оно не соответствует допустимым, возвращать ‘none’.

КОД ПРОГРАММЫ

***main.cpp***

|  |
| --- |
|  |
|  | #include <iostream>  #include <vector>  #include "histogram.h"  #pragma hdrstop  #include "svg.h"  #include "func.h"  using namespace std;        int main()  {  foo(2);  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,number\_count);      string text\_decoration;  cerr << "Enter text-decoration: ";  cin >> text\_decoration;  decor(text\_decoration);    show\_histogram\_svg(bins, text\_decoration);  return 0;  } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***histogram.h***

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

***histogram.cpp***

|  |
| --- |
| #include "histogram.h" |
|  | #include "histogram.h"  #include <vector>  #include <iostream>  #include<conio.h>    using namespace std;    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;  }  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, double bin\_count,double number\_count)  {    double max, min;  find\_minmax(numbers, min, max);  double bin\_size = (max - min) / bin\_count;  vector<size\_t> bins(bin\_count);  for(int i = 0; i < number\_count; i++)  {  bool found = false;  for (size\_t j = 0; (j < bin\_count - 1) && !found; j++)  {  auto lo = min + j \* bin\_size;  auto hi = min + (j + 1) \* bin\_size;  if ((lo <= numbers[i]) && (numbers[i] < hi))  {  bins[j]++;  found = true;  }  }  if (!found)  {  bins[bin\_count - 1]++;  }  }  return bins;  } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***SVG.h***

|  |
| --- |
| #ifndef SVG\_H\_INCLUDED |
|  | #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 decor(string &text\_decoration);    void show\_histogram\_svg(const vector<size\_t>& bins, string text\_decoration);    #endif // SVG\_H\_INCLUDED |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***SVG.cpp***

|  |
| --- |
| #include "SVG.h" |
|  | #include "SVG.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,string text\_decoration)  {  cout << "<text x='" << left << "' y='"<< baseline <<"' text-decoration='" <<text\_decoration << "'>"<<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,string text\_decoration)  {  const auto IMAGE\_WIDTH = 400;  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 = 10;  const auto MAX\_COUNT = (IMAGE\_WIDTH-TEXT\_WIDTH)/ BLOCK\_WIDTH;    double top = 0;  svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);  size\_t count;  size\_t max\_count = bins[0];  for (size\_t bin : bins)  {  if (bin > max\_count)  {  max\_count = bin;  }  }    if (max\_count > MAX\_COUNT)  {    const double h = (double)MAX\_COUNT / max\_count;    for (size\_t bin : bins)  {  auto height = (size\_t)(bin \* h);  const double bin\_width = BLOCK\_WIDTH \* height;  svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin),text\_decoration);  svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "black", "red");  top += BIN\_HEIGHT;    }    svg\_end();  }  else  {  for (size\_t bin : bins)  {    const double bin\_width = BLOCK\_WIDTH \* bin;  svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin),text\_decoration);  svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "black", "red");  top += BIN\_HEIGHT;    }  svg\_end();    }      }  void decor(string &text\_decoration)  {  if (text\_decoration!="underline" & text\_decoration!="line-through" & text\_decoration!="none" & text\_decoration!="overline")  {  text\_decoration="none";  }  return ;  } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***test.cpp***

|  |
| --- |
| #include "histogram.h" |
|  | #include "histogram.h"  #include "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\_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\_indiv1()  {  string text\_decoration="underline";  decor(text\_decoration);  assert(text\_decoration == "underline");  }  void  test\_indiv2()  {  string text\_decoration="underlin";  decor(text\_decoration);  assert(text\_decoration == "none");  }  void  test\_indiv3()  {  string text\_decoration="none";  decor(text\_decoration);  assert(text\_decoration == "none");  }  int main()  {  test\_positive();  test\_negative();  test\_same();  test\_onenumber();  test\_empty();  test\_indiv1();  test\_indiv2();  test\_indiv3();  } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |