**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

Московский институт электроники и математики Национального

исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

**ОТЧЕТ**

**По лабораторной работе №1**

**По курсу «Алгоритмизация и программирование»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | ФИО студента | | Номер группы | Дата |
| Псурцев Иван Вадимович | БПМ-192 | 03.10.2020 |
|  |
|  |
|  |

**Москва – 2020 г.**

**ЗАДАНИЕ (вариант №14)**

На основе примера Histogrammer\_vs2013.zip выполнить следующие задания: (При работе с Visual Studio 2013 или новее в Windows распаковать архив и дважды кликнуть на файл с расширением sln. При работе не в Windows или в случае проблем с открытием проекта создать новый проект в редакторе и добавить в него файлы с расширением cpp, h и inl)

2. Создать функцию SortByVal для сортировки столбцов гистограммы по их величине по возрастанию (по умолчанию) и убыванию в зависимости от передаваемого параметра bAscending. Интерфейс: void SortByVal (Histogram &historgam, bool bAscending = true);

5. Изменить функцию PrintHistogram таким образом, чтобы она работала корректно, в случае, если имена столбцов имеют разную длину (все столбцы печатаются с одинаковым отступом от края экрана, имена столбцов отформатированы по левому краю).

F. Написать функцию построения гистограммы, иллюстрирующей частоту встречающихся символов в передаваемой строке, не считая разделителей.

**РЕШЕНИЕ**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <algorithm>

#include "histogram.h"

void SortByVal(Histogram& historgam, bool bAscending = true);

bool ComparePairGreater(std::pair<int, char\*> l, std::pair<int, char\*> r);

bool ComparePairLess(std::pair<int, char\*> l, std::pair<int, char\*> r);

char\* RandString(int& cmpMaxLen, int strMinLen = 1, int strMaxLen = 8);

void AddRandomBlocks(Histogram& hist);

int main()

{

std::srand(time(NULL));

const char\* const strProgramName = "Histogramer 1.1";

std::cout << std::setfill('\*') << std::setw(strlen(strProgramName) + 4) << "" << std::endl;

std::cout << '\*' << std::setfill(' ') << std::setw(strlen(strProgramName) + 3) << '\*' << std::endl;

std::cout << "\* " << strProgramName << " \*" << std::endl;

std::cout << '\*' << std::setfill(' ') << std::setw(strlen(strProgramName) + 3) << '\*' << std::endl;

std::cout << std::setfill('\*') << std::setw(strlen(strProgramName) + 4) << "" << std::endl << std::endl;

Histogram myBarChart;

InitHistogram(myBarChart);

AddRandomBlocks(myBarChart);

/\*AddBlock(myBarChart, "a");

AddBlock(myBarChart, "a");

AddBlock(myBarChart, "a");

AddBlock(myBarChart, "b");

AddBlock(myBarChart, "c");

AddBlock(myBarChart, "a");

AddBlock(myBarChart, "d");

AddBlock(myBarChart, "e");

AddBlock(myBarChart, "c");

AddBlock(myBarChart, "z");

AddBlock(myBarChart, "z");

AddBlock(myBarChart, "z");

AddBlock(myBarChart, "z");

AddBlock(myBarChart, "z");

AddBlock(myBarChart, "Y");

AddBlock(myBarChart, "Y");\*/

std::cout << "Before sorting: \n";

PrintHistogram(myBarChart);

SortByVal(myBarChart, false);

std::cout << "\nAfter sorting: \n";

PrintHistogram(myBarChart);

DestroyHistogram(myBarChart);

Histogram strHist;

InitHistogram(strHist);

int strLength = 0;

char\* randomString = RandString(strLength, 16, 32);

for (int i = 0; i < strLength; ++i)

{

char\* tmpStr = (char\*)calloc(2, sizeof(char));

tmpStr[0] = randomString[i];

AddBlock(strHist, tmpStr);

}

std::cout << "\nString: " << randomString << std::endl;

SortByVal(strHist);

PrintHistogram(strHist);

DestroyHistogram(strHist);

return 0;

}

char\* RandString(int& cmpMaxLen, int strMinLen, int strMaxLen)

{

char length = rand() % strMaxLen + strMinLen;

if (length > cmpMaxLen)

{

cmpMaxLen = length;

}

char\* str = (char\*)calloc(length + 1, sizeof(char));

for (int i = 0; i < length; ++i)

{

str[i] = 97 + rand() % 25;

}

return str;

}

void AddRandomBlocks(Histogram& hist)

{

char count = 4 + rand() % 10;

for (char i = 0; i < count; ++i)

{

char\* str = RandString(hist.maxLength);

char size = 1 + rand() % 15;

for (char j = 0; j < size; ++j)

{

AddBlock(hist, str);

}

}

}

bool ComparePairGreater(std::pair<int, char> l, std::pair<int, char> r)

{

return l.first < r.first;

}

bool ComparePairLess(std::pair<int, char\*> l, std::pair<int, char\*> r)

{

return l.first > r.first;

}

void SortByVal(Histogram& historgam, bool bAscending)

{

std::pair<int, char\*>\* pairArray = (std::pair<int, char\*>\*)calloc(historgam.nBarsSize, sizeof(std::pair<int, char\*>));

for (int i = 0; i < historgam.nBarsSize; ++i)

{

pairArray[i] = std::pair<int, char\*>(historgam.Bars[i], historgam.BarNames[i]);

}

bAscending ? std::sort(pairArray, pairArray + historgam.nBarsSize, ComparePairGreater) : std::sort(pairArray, pairArray + historgam.nBarsSize, ComparePairLess);

for (int i = 0; i < historgam.nBarsSize; ++i)

{

historgam.Bars[i] = pairArray[i].first;

historgam.BarNames[i] = pairArray[i].second;

}

}

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "histogram.h"

#include <iostream>

#include <iomanip>

#include <algorithm>

using namespace std;

///////////////////////////////////////////////////////////////////

// Вспомогательные функции

void AddBar(Histogram &historgam, const char \*strName);

bool AppendBar(Histogram &historgam, const char \*strName);

///////////////////////////////////////////////////////////////////

// реализация функций из файла histogram.h

//-----------------------------------------------------------------

// Функция AddBlock увеличивает значение столбца strName гистограмы historgam на 1.

// Если столбца с таким именем не существует, он создается.

void AddBlock(Histogram &historgam, const char \*strName){

if(!AppendBar(historgam, strName)){

AddBar(historgam, strName);

}

}

//-----------------------------------------------------------------

// Функция PrintHistogram выводит гистограму на экран.

void PrintHistogram(const Histogram &histogram){

for(int i=0; i<histogram.nBarsSize; i++){

cout << histogram.BarNames[i];

for (int j = 0; j < histogram.maxLength - strlen(histogram.BarNames[i]); ++j)

{

cout << ' ';

}

cout << '|';

//cout.setf

cout << setfill(histogram.chBlock) << setw(10\*histogram.Bars[i]/histogram.nMaxBar) << "" << /\*ends <<\*/ endl;

}

}

///////////////////////////////////////////////////////////////////

// реализация вспомогательных функций

//-----------------------------------------------------------------

// Функция AppendBar увеличивает значение столбца strName гистограмы historgam на 1.

bool AppendBar(Histogram &historgam, const char \*strName){

for(int i=0; i<historgam.nBarsSize; i++){

if(strcmp(historgam.BarNames[i], strName)==0){

historgam.nMaxBar = max(historgam.nMaxBar, ++historgam.Bars[i]);

return true;

}

}

return false;

}

// Функция AddBar создает в гистограме historgam новый столбец с именем strName.

//-----------------------------------------------------------------

void AddBar(Histogram &historgam, const char \*strName){

char \*\*newNames = new char \*[historgam.nBarsSize+1];

int \*newBars = new int[historgam.nBarsSize+1];

if(historgam.nBarsSize > 0){

for(int i=0; i<historgam.nBarsSize; i++){

newNames[i] = historgam.BarNames[i];

newBars[i] = historgam.Bars[i];

}

delete[] historgam.BarNames;

delete[] historgam.Bars;

}

newNames[historgam.nBarsSize] = new char[strlen(strName)+1];

historgam.maxLength = std::max(static\_cast<int>(strlen(strName)), historgam.maxLength); // Проверяем на максимальную длину строки

strcpy(newNames[historgam.nBarsSize], strName);

newBars[historgam.nBarsSize] = 1;

historgam.nBarsSize++;

historgam.BarNames = newNames;

historgam.Bars = newBars;

}

#pragma once

//Структура для описания гисторгаммы

typedef struct{

char \*\*BarNames;

int \*Bars;

int nBarsSize;

char chBlock;

int nMaxBar;

int maxLength;

} Histogram;

//Функции для работы с гисторгаммой

void inline InitHistogram(Histogram &historgam);

void inline DestroyHistogram(Histogram &historgam);

void PrintHistogram(const Histogram &historgam);

void AddBlock(Histogram &historgam, const char \*strName);

#include "histogram.inl"

#include "histogram.cpp"

//-----------------------------------------------------------------

// Функция InitHistogram инициализирует данные гистограммы

void inline InitHistogram(Histogram &historgam){

historgam.nBarsSize = 0;

historgam.BarNames = 0;//Всегда обнуляйте неинициализированные указатели

historgam.Bars = 0; //Всегда обнуляйте неинициализированные указатели

historgam.chBlock = '#';

historgam.nMaxBar = 1;

}

//-----------------------------------------------------------------

// Функция DesroyHistogram освобождает занятую память

void inline DestroyHistogram(Histogram &historgam){

if(historgam.nBarsSize){

delete[] historgam.BarNames;

historgam.BarNames = 0; //Всегда обнуляйте указатели после освобождения памяти

delete[] historgam.Bars;

historgam.Bars = 0; //Всегда обнуляйте указатели после освобождения памяти

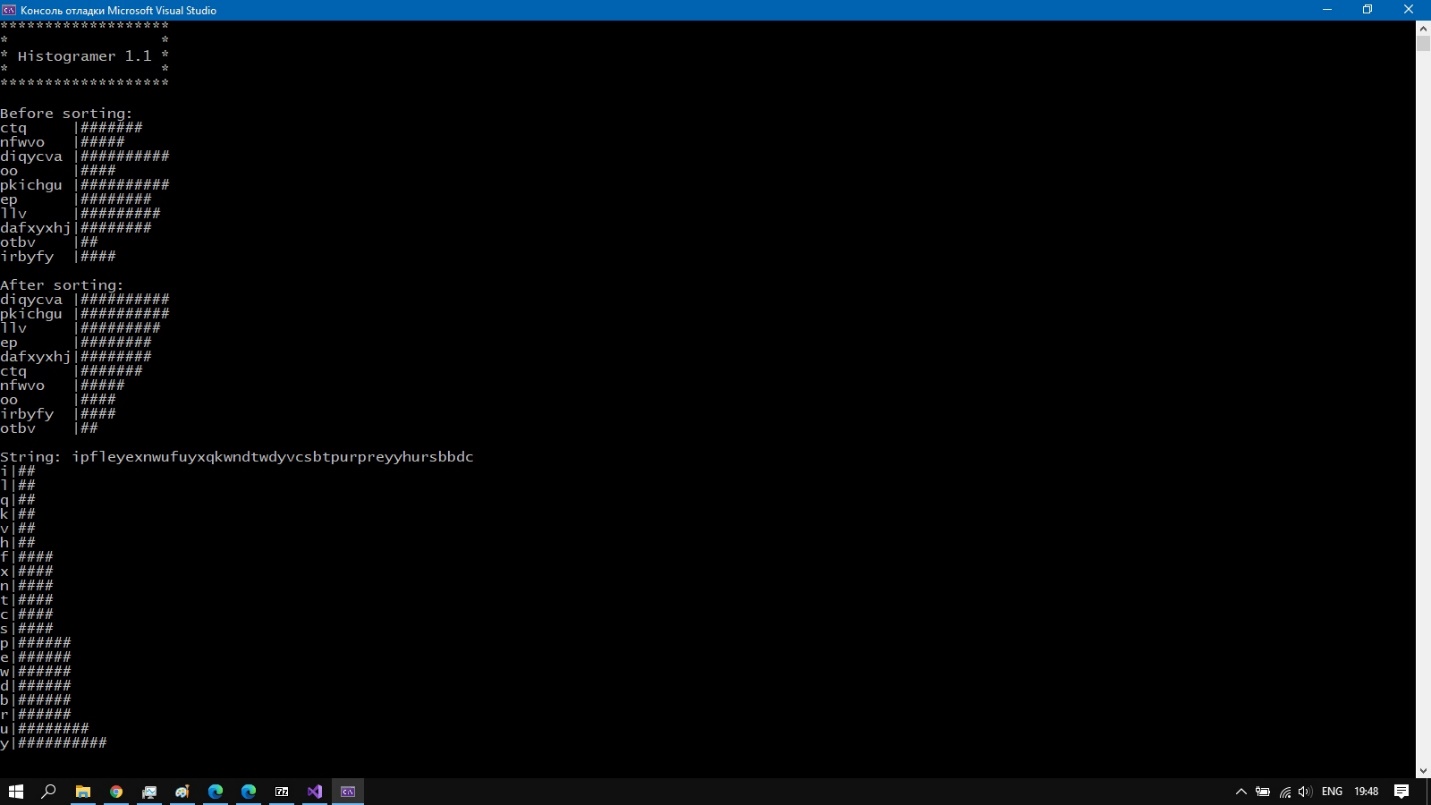
}

historgam.nMaxBar = 1;

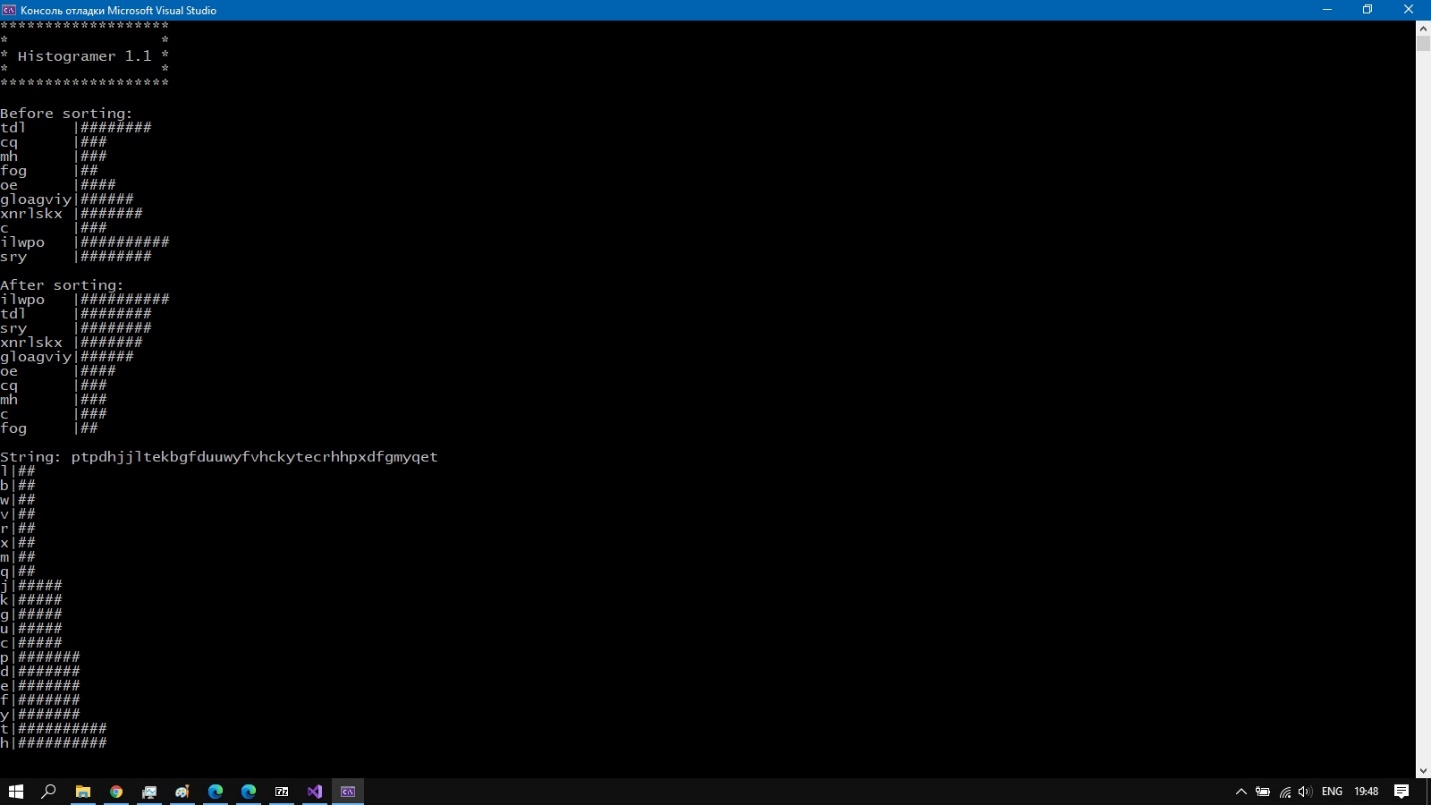
}

**ТЕСТЫ**

**Тест № 1**



**Тест № 2**



**Тест №3**

