IPI-Zettel 6

Jan Maintok, Raphael Senghaas, Kianusch Vahid Yousefnia
Tutor: Enes Witwit

Aufgabe 1

```cpp
//And possible improvement of the excercise:
//Include a number x with -1 < x < 1 and abs(x) > 0.005 in the vector numbers
//a number of this format can cause some problems for some programs

//Include std header
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <cmath>

//Include self-defined header
#include "format_numbers.hpp"

/*Declaration of Functions*/
int double_to_int(double number); //Convert double to integer
double round_two_digits(double number); //Round a double to the second digit

std::string double_to_string(double x); //convert a double to a string

std::vector<int> vec_to_int(std::vector<double> v); //convert a double vector to
     an integer vector
std::vector<double> vec_rounded(std::vector<double> v); //Convert a vector of
    doubles of an vector of double rounded to two digits
std::vector<std::string> format_numbers(std::vector<double> v); //Convert a
    vector to the format that is asked for in the excersice

int main()
{
    std::vector<double> double_vector = { -1.676, 19.557, 2.255, 1.527, 36.345};
     //define a test vector of doubles
    //Output of test vector
    for (double n : double_vector)
    {
        std::cout << n << " ";
    }
    std::cout << std::endl;
    //Covert test vector to integer vector
    std::vector<int> int_vector = vec_to_int(double_vector);

    //Output of the rounded test values
    for (int n : int_vector)
    {
        std::cout << n << " ";
    }
    std::cout << std::endl;

    //Convert vector to a vector rounded to two digits
    std::vector<double> double_vector_2 = vec_rounded(double_vector);
    //Outout of this vector
    for (double n : double_vector_2)
    {
        std::cout << n << " ";
    }
    std::cout << std::endl << std::endl;

    //Output from test vector from format_numbers.hpp
    for (double n : numbers)
    {
        std::cout << n << " ";
    }
    //Convert numbers vector to vector of formated strings
    std::vector<std::string> string_vector = format_numbers(numbers);
    std::cout << std::endl << std::endl;

    //Output of this vector
    for (std::string n : string_vector)
    {
        std::cout << n << std::endl;
    }
    return 0;
}

int double_to_int(double number)
{
    return static_cast<int>(number + 0.5); //Round by just take the integer part
     of 1/2 + number
}

double round_two_digits(double number)
{
    if (number == 0) //because later the number is devide by number, check
    whether number == 0 zero (alternativ: if(!number){...}
    {
        return 0.;
    }
    return  (std::abs(number) / number) * static_cast<long>(std::abs(number) *
    100 + 0.5) / 100.0; //Round number to two digits
}

std::string double_to_string(double x)
{
    x = round_two_digits(x); //Round the input number x
    if (x == 0)               //check whether x is zero
    {
        return "            0.00";
    }
    std::string ret_string = std::to_string(x); //Convert  x to a string
    ret_string += "00";  //add "00" to the end to garantee that there are 2 digits after the decimal point
    //The numbers x of with - 1 < x < 1 and abs(x) > 0.005 causes some trouble, that's why they are treated extra here
    //For abs(x) > 0.005 is round_two_digits(x) = 0. Hence x satisfies x == 0 and is already treated above.
    if (x < 0 && x > -1)
    {
        ret_string.resize(5);
    }
    else if (x < 1 && x > 0)
    {
        ret_string.resize(4);
    }
    //resize the number in a way that exactly two digits after the decimal point are left
    else
    {
        ret_string.resize(static_cast<int>(std::log10(std::abs(x))) + 4 + (x - std::abs(x)) / (2 * x));
    }

    //Insert ' (an apostroph) every 3 digits
    for (unsigned int i = 1; i < ret_string.length() - 3; i++)
    {
        if ((ret_string.length() - i) % 3 == 0 && ret_string[i-1] != '-')
        {
            ret_string.insert(i, 1, '\'');
            i++;
        }
    }

    //Insert " " infront of the actual number until the length of the string is >= 16
    while (ret_string.size() < 16)
    {
        ret_string = " " + ret_string;
    }
    //garantee that the size of the string is equal to 16
    ret_string.resize(16);

    return ret_string;
}

//tranform all numbers in a vector to integer
std::vector<int> vec_to_int(std::vector<double> v)
{
    //initialize the vector we want to return
    std::vector<int> ret_vector;
    ret_vector.resize(v.size());

    //Use transform with lambda function double to int to convert v to an int_vector and save to ret_vector
    std::transform(v.begin(), v.end(), ret_vector.begin(), double_to_int);

    return ret_vector;
}
//round all numbers in a vector to two digits
std::vector<double> vec_rounded(std::vector<double> v)
{
    //initialize the vector we want to return
    std::vector<double> ret_vector;
    ret_vector.resize(v.size());

    //Use transform with lambda function round_two_digits to convert v to an double_vector where all numbers are
    //rounded to two digits and save to ret_vector
    std::transform(v.begin(), v.end(), ret_vector.begin(), round_two_digits);

    return ret_vector;
}


//Convert an vector v of doubles to a vector of strings
std::vector<std::string> format_numbers(std::vector<double> v)
{
    //initialize the vector we want to return
    std::vector<std::string> ret_vector;
    ret_vector.resize(v.size());

    //Use transform with lambda function double_to_string
    std::transform(v.begin(), v.end(), ret_vector.begin(), double_to_string);

    return ret_vector;
}
```

Aufgabe 2

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <string>
#include "sort_versions.hpp"
#include <algorithm>

using namespace std;

/*
function to turn version-string into a vector with the numbers
```

```
in the respective places
*/
vector<int> split_version(string version){
  // develop current number
  string current_number = "";
  vector<int> output;
  int i = 0;
  while (i < version.length()){
    if (version[i] != '.'){
      current_number += version[i];
    }
    else {
      output.push_back(atoi(current_number.c_str()));
      current_number = "";
    }
    i++;
  }
  output.push_back(atoi(current_number.c_str())); // Add last number to output
  output.push_back(-1); // Add -1 to the end, to make the program see 1.1.1
     bigger than 1.1
  return output;
}


/*
function, that returns true if version v1 is smaller than version v2
*/
bool version_less(string v1, string v2){
  vector<int> vers1 = split_version(v1);
  vector<int> vers2 = split_version(v2);

  //going through the levels of the version number
  if (vers1.size() <= vers2.size())
    int size = vers1.size();
  else
    int size = vers2.size();
  for (int i=0; i<vers1.size(); i++){
    if (vers1[i] < vers2[i])
      return true;
    else if (vers1[i] > vers2[i])
      return false;
  }
  return false;
}



int main(){
  sort(versions.begin(), versions.end(), version_less);
  cout << "Versions: " << endl;
  for (auto it=versions.begin(); it!=versions.end(); ++it)
    cout << *it << endl;

  return 0;
}
```

Aufgabe 3

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <vector>
using namespace std;

//Declarations


int main (){
  // a) Read in encrypted file
  ifstream infile("encrypted_text.txt");
  string text;
  string line;
  while (infile){
    getline(infile,line);
    text += line + "\n";
  }

  // b) count frequency of letters
  map<char, int> counts;
  for (char & current : text){
    if (isalpha(current)){
      char current_lower = tolower(current); //treat upper case letters like lowers
      counts[current_lower]++;
    }

  }


  // c) sort counts in an ascending way by switching key and value of counts
  map<int, char> sorted;
  for (auto iter = counts.begin(); iter != counts.end(); iter++){
    sorted[(*iter).second] = (*iter).first;
  }

  // frequency of english letters
  vector<char> letters = {'z', 'j', 'q', 'x', 'k', 'v', 'b', 'y',
  'g', 'p', 'w', 'f', 'm', 'c', 'u', 'l', 'd', 'r', 'h', 's',
  'n', 'i', 'o', 'a', 't', 'e'};

  // d) decryption table: match the k-th-most frequently found letter to letters[k]
  map<char, char> decrypt;
```

```cpp
  int k = 0;
  for (auto iter = sorted.begin(); iter != sorted.end(); iter++){
    char current_letter = (*iter).second;
    //create entry in decrypt for lower- and upper-case letter
    decrypt[current_letter] = letters[k];
    decrypt[toupper(current_letter)] = toupper(letters[k]);
    k++;
  }

  // e) decryption of given text and creation of output file
  for (char & iter : text){
    if (isalpha(iter)){
      iter = decrypt[(iter)];
    }
  }

  ofstream outfile("decrypted_text.txt");
  outfile << text;


  return 0;
}
```