Лабораторная работа №1

Селезнев Василий Александрович - студент группы НПМмд-02-22

16.09.2022

Шифры простой замены

Прагматика выполнения

Умение пользоваться шифрами Цезаря и Атбаша

Цель выполнения лабораторной работы

Освоить на практике использование шифров Цезаря и Атбаша

Задачи выполнения работы

Написать функции, которые реализуют шифрование шифрами Цезаря и Атбаша

Результаты выполнения лабораторной работы. Часть 1

```cpp
#include "../include/CipherHelper.h"

#include <iostream>

//const std::string CipherHelper::engAlphabetLower = "abcdefghijklmnopqrstuvwxyz";
const std::string CipherHelper::engAlphabetUpper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ";

void CipherCaesar::cipher(const std::string& message, int Key, std::string& encrypted){
    // char with index j => char with index (j + k) / mod 26
    if (!encrypted.empty()){
        throw std::invalid_argument( s: "encrypted is not empty!" );
    }

    if (!message.empty()){
        for (auto character :char  : message){
            // test it, if it returns right index
            auto index =  = engAlphabetUpper.find( c: character);
            encrypted += engAlphabetUpper[(index + Key) % engAlphabetUpper.size()];
        }
    }
    else{
        encrypted = "";
    }
}
```

Figure 1: Caesar cipher

Написал код для дешивровки кодов шифром Цезаря

```cpp
void CipherCaesar::decipher(const std::string &message, int Key, std::string &decrypted) {
    if (!decrypted.empty()){
        throw std::invalid_argument( s: "decrypted is not empty!" );
    }

    if (!message.empty()){
        for (auto character :char : message){
            auto index :  = engAlphabetUpper.find( c: character);
            decrypted += engAlphabetUpper[ (engAlphabetUpper.size() +  (index - Key)) % engAlphabetUpper.size() ];
        }
    }
    else{
        decrypted = "";
    }
}
```

Figure 2: Caesar decipher

Написал код для зашивровки кодов шифром Атбаша

```cpp
void CipherAtbash::cipher(const std::string &message, std::string &encrypted) {
    if (!encrypted.empty()){
        throw std::invalid_argument( s: "encrypted is not empty!" );
    }

    if (!message.empty()){
        for (auto character :char  : message){
            // test it, if it returns right index
            auto index :  = engAlphabetUpper.find( c: character);
            encrypted += engAlphabetUpper[engAlphabetUpper.size() - 1 - index];
        }
    }
    else{
        encrypted = "";
    }
}
```

Figure 3: Atbash cipher

Написал код для дешивровки кодов шифром Атбаша

```
void CipherAtbash::decipher(const std::string &message, int Key, std::string &decrypted) {
    if (!decrypted.empty()){
        throw std::invalid_argument( s: "decrypted is not empty!" );
    }
```

Figure 4: Atbash decipher

Написал заголовочный файл для класса реализации CipherHelper

```cpp
#ifndef LAB01_CIPHERHELPER_H
#define LAB01_CIPHERHELPER_H

#include <string>


class CipherHelper{
public:
    static const std::string engAlphabetLower;
    static const std::string engAlphabetUpper;
};

class CipherCaesar : CipherHelper {
public:
    static void cipher (const std::string& message, int Key, std::string& encrypted);
    static void decipher (const std::string& message, int Key, std::string& decrypted);
};

class CipherAtbash : CipherHelper {
public:
    static void cipher (const std::string& message, std::string& encrypted);
//    static void decipher (const std::string& message, int Key, std::string& decrypted);
};



#endif //LAB01_CIPHERHELPER_H
```

Figure 5: Header file

Написал CMakeLists.txt файл, который создаёт библиотеку из класса CipherHelper и бинарник main

```
cmake_minimum_required(VERSION 3.20)
project(lab01)

set(CMAKE_CXX_STANDARD 14)

add_library(lab01 src/CipherHelper.cpp)

add_executable(main src/main.cpp)
target_link_libraries(main lab01)
```

Figure 6: CmakeLists.txt file

Написал main.cpp файл, в котором есть тесты реализованных функций.

Часть шифра Цезаря:

```cpp
int main(){

    // Caesar part

    std::string msg1 = "MY NAME IS VASYA";
    std::string msg2 = "I LOVE STUDY";
    std::string msg3 = "RUDN IS THE BEST UNIVERSITY";

    std::string enc1 = "", enc2 = "", enc3 = "";
    std::string dec1 = "", dec2 = "", dec3 = "";

    CipherCaesar::cipher( message: msg1,  Key: 3,  &: enc1);
    CipherCaesar::cipher( message: msg2,  Key: 3,  &: enc2);
    CipherCaesar::cipher( message: msg3,  Key: 3,  &: enc3);

    CipherCaesar::decipher( message: enc1,  Key: 3,  &: dec1);
    CipherCaesar::decipher( message: enc2,  Key: 3,  &: dec2);
    CipherCaesar::decipher( message: enc3,  Key: 3,  &: dec3);

    std::cout << enc1 << std::endl;
    std::cout << enc2 << std::endl;
    std::cout << enc3 << std::endl << std::endl;

    std::cout << dec1 << std::endl;
    std::cout << dec2 << std::endl;
    std::cout << dec3 << std::endl << std::endl;
```

Часть шифра Атбаша:

```
    enc1 = "", enc2 = "", enc3 = "";
    dec1 = "", dec2 = "", dec3 = "";

    CipherAtbash::cipher( message: msg1,  &: enc1);
    CipherAtbash::cipher( message: msg2,  &: enc2);
    CipherAtbash::cipher( message: msg3,  &: enc3);

    std::cout << enc1 << std::endl;
    std::cout << enc2 << std::endl;
    std::cout << enc3 << std::endl << std::endl;

    CipherAtbash::cipher( message: enc1,  &: dec1);
    CipherAtbash::cipher( message: enc2,  &: dec2);
    CipherAtbash::cipher( message: enc3,  &: dec3);

    std::cout << dec1 << std::endl;
    std::cout << dec2 << std::endl;
    std::cout << dec3 << std::endl << std::endl;



}
```

Особенности реализации

- Реализовано на C++, поэтому код быстрый
- Функции статические, поэтому экономятся ресурсы. Особенно это актуально если нам нужно шифровать примерно 1000 сообщений в секунду. Для этих 1000 сообщений не будут создаваться 1000 экземпляров классов
- Все собрано в проект CMake, поэтому код кросс-платформенный

Выводы

Освоил на практике применение шифрования шифрами Цезаря и Атбаш.