Министерство науки И ВЫСШЕГО ОБРАЗОВАНИЯ   
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Национальный исследовательский Томский политехнический Университет»



Инженерная школа автоматизации и робототехники

Отделение автоматизации и робототехники

15.04.04 «Автоматизация технологических процессов и производств»

Вариант 1

Лабораторная работа №3

**«Классические криптографические системы»**

по дисциплине:

**«Информационная безопасность автоматизированных систем»**

**Исполнитель:**

студент группы 8ТМ22 Гао Аоцзе \_\_\_\_\_\_\_\_\_\_

**Руководитель:**

К.т.н., доцент ОАР Суходоев Михаил Сергеевич \_\_\_\_\_\_\_\_\_\_

Томск – 2023

## Цель работы

Изучить классические методы шифрования и программно реализовать алгоритмы шифрования текстовых сообщений.

## Задачи

1. Выбрать любые два классических криптоалгоритма из представленных в методических указаниях.
2. Реализовать программу для шифрования и дешифрования текстовых сообщений. Программа должна осуществлять шифрование текста, введенного пользователем, с помощью выбранных алгоритмов. При необходимости указания параметров шифрования (например, сдвига в алгоритме Цезаря) пользователь должен также вводить их в командную строку в ответ на соответствующий запрос. Оформить классы в виде dll библиотек.

3. Осущест вить проверку правильности работы программы. Для этого необходимо вручную зашифровать некоторое сообщение в соответствии с выбранным алгоритмом и сравнить полученный результат с результатом работы программы. В качестве сообщения используйте Вашу фамилию, записанную латиницей.

4. Оформить отчет по лабораторной работе.

## Ход работы

**Шифр Цезаря**

В I в. н.э. Ю.Цезарь во время войны с галлами, переписываясь со своими друзьями в Риме, заменял в сообщении первую букву латинского алфавита (*А*) на четвертую (*D*), вторую (*B)* – на пятую (*E*), наконец последнюю – на третью:

**ABCDEFGHIJKLMNOPQRSTUVWXYZ YHQL YLGL YLFL**

**DEFGHIJKLMNOPQRSTUVWXYZABC Veni vidi vici -** “Пришел, увидел,

победил”

С современной точки зрения, в шифре Цезаря каждая буква замещается на букву, находящуюся *k* символами правее по модулю равному количеству букв в алфавите.

*Ck*(*j*) *=*(*j+k*)(mod *n*),

где *n* *—* количество букв в алфавите (мощность алфавита).

Очевидно, что обратной подстановкой является

*Ck*-1(*j*) *= Сn-k=*(*j*+*n*-*k*)(mod *n*).

**Аффинная криптосистема**

Обобщением системы Цезаря является аффинная криптосистема. Она определяется двумя числами *a* и *b*, где ***0 <=*** *a,b* <= *n*-1 и *n* —является мощностью алфавита. Числа *a* и *n* должны быть взаимно простыми. Соответствующими заменами являются: *Aa,b*(*j*)=(*a\*j+b*)(mod *n*),

Обратную замену также можно получить, просто поменяв местами строки в таблице замен.

*A*-1*a,b*(*j*)=(*j-b*)\**a*-1(mod *n*).

Взаимная простота *a* и *n* необходима для однозначности отображения, в противном случае возможны отображения различных символов в один и неоднозначность дешифрирования.

В данном проекте мы выберем шифр Цезаря и шифр Имитации, а затем создадим dll.Шаги для запуска программы следующие:

1. Программа запускается, выводит меню и выбирает функцию, которую нужно использовать: шифрование, дешифрование.

2. вводим используемый алгоритм: шифрование шифра Цезаря, дешифрование шифра Цезаря, шифрование шифра Имитации, дешифрование шифра Имитации.

3. вводим текст для шифрования/дешифрования и параметры шифрования.

4. Вывести результат и вернуться в начальное меню.

Устройство, используемое в этом тесте, - MacBook, среда программирования - Xcode, DLL (Dynamic Link Library) написана на C++, а тестовая программа - на C++ в Xcode.

Создание библиотеки динамических связей

Открываем Xcode, выбираем create New Project > Library in Framework&Library, вводим название продукта "mylib", выбираем язык C++, выбираем Dynamic, после чего интерфейс покажет, что проект создан. Кликаем правой кнопкой мыши на проекте, создаем новый файл, выбираем C++ и даем имя файлу. После создания в поле проекта появятся заголовочные и исходные файлы. Приступим к программированию.

Прежде всего, делаем объявление в заголовочном файле:

#include <string>

**extern** "C" **const** **char**\* caesar\_encrypt(**const** **char**\* plain\_text, **int** offset);

**extern** "C" **const** **char**\* caesar\_decrypt(**const** **char**\* cipher\_text, **int** offset);

// 仿射密码

**extern** "C" **const** **char**\* affine\_encrypt(**const** **char**\* plain\_text, **int** a, **int** b);

**extern** "C" **const** **char**\* affine\_decrypt(**const** **char**\* cipher\_text, **int** a, **int** b);

Следующий шаг - написать функции шифрования и дешифрования:

// 凯撒密码

**extern** "C" **const** **char**\* caesar\_encrypt(**const** **char**\* plain\_text, **int** offset) {

    std::string cipher\_text;

**for** (**const** **char**\* p = plain\_text; \*p; ++p) {

        cipher\_text += **char**((\*p - 'a' + offset) % 26 + 'a');

    }

**return** strdup(cipher\_text.c\_str());

}

**extern** "C" **const** **char**\* caesar\_decrypt(**const** **char**\* cipher\_text, **int** offset) {

    std::string plain\_text;

**for** (**const** **char**\* p = cipher\_text; \*p; ++p) {

        plain\_text += **char**((\*p - 'a' - offset + 26) % 26 + 'a');

    }

**return** strdup(plain\_text.c\_str());

}

// 仿射密码

**extern** "C" **const** **char**\* affine\_encrypt(**const** **char**\* plain\_text, **int** a, **int** b) {

    std::string cipher\_text;

**for** (**const** **char**\* p = plain\_text; \*p; ++p) {

**if** ('a' <= \*p && \*p <= 'z') {

               cipher\_text += **char**((a \* (\*p - 'a') + b) % 26 + 'a');

           } **else** {

               // 如果字符不是小写字母，直接保留

               cipher\_text += \*p;

           }

       }

**return** strdup(cipher\_text.c\_str());

}

**extern** "C" **const** **char**\* affine\_decrypt(**const** **char**\* cipher\_text, **int** a, **int** b) {

    std::string plain\_text;

**int** a\_inv = -1;

**for** (**int** i = 0; i < 26; ++i) {

           // 寻找加法逆元

**if** ((a \* i) % 26 == 1) {

               a\_inv = i;

**break**;

           }

       }

**if** (a\_inv == -1) {

           // 没有找到加法逆元，无法解密

**return** strdup("");

       }

**for** (**const** **char**\* p = cipher\_text; \*p; ++p) {

**if** ('a' <= \*p && \*p <= 'z') {

               plain\_text += **char**((a\_inv \* (\*p - 'a' - b + 26)) % 26 + 'a');

           } **else** {

               // 如果字符不是小写字母，直接保留

               plain\_text += \*p;

           }

       }

**return** strdup(plain\_text.c\_str());

}

После выполнения описанных выше действий выберите build, чтобы создать DLL.

Далее мы напишем тестовое приложение, вызывающее DLL и создающее меню для реализации функциональности, ориентированной на пользователя.

Для вызова DLL мы можем использовать файл dlfcn.h.

//  main.cpp

//  Test1712

//  Created by 高澳杰 on 2023/12/17.

//

#include <iostream>

#include <dlfcn.h>

#include <cstring>

**int** main() {

    // 1. Load the dynamic library

**void**\* libHandle = dlopen("/Users/gaoaojie/Desktop/Debug/Test1712/libmylib1712.dylib", RTLD\_LAZY);

    // Check if the library loaded successfully

**if** (!libHandle) {

        std::cerr << "Error loading the library: " << dlerror() << std::endl;

**return** 1;

    }

    // 2. Get function pointers

**using** CaesarEncryptFunc = **const** **char**\* (\*)(**const** **char**\*, **int**);

**using** CaesarDecryptFunc = **const** **char**\* (\*)(**const** **char**\*, **int**);

**using** AffineEncryptFunc = **const** **char**\* (\*)(**const** **char**\*, **int**, **int**);

**using** AffineDecryptFunc = **const** **char**\* (\*)(**const** **char**\*, **int**, **int**);

    CaesarEncryptFunc caesarEncrypt = **reinterpret\_cast**<CaesarEncryptFunc>(dlsym(libHandle, "caesar\_encrypt"));

    CaesarDecryptFunc caesarDecrypt = **reinterpret\_cast**<CaesarDecryptFunc>(dlsym(libHandle, "caesar\_decrypt"));

    AffineEncryptFunc affineEncrypt = **reinterpret\_cast**<AffineEncryptFunc>(dlsym(libHandle, "affine\_encrypt"));

    AffineDecryptFunc affineDecrypt = **reinterpret\_cast**<AffineDecryptFunc>(dlsym(libHandle, "affine\_decrypt"));

    // Check if function pointers are obtained successfully

**if** (!caesarEncrypt || !caesarDecrypt || !affineEncrypt || !affineDecrypt) {

        std::cerr << "Error getting function pointers: " << dlerror() << std::endl;

        dlclose(libHandle);

**return** 1;

    }

**while** (**true**) {

        // 3. Display menu

        std::cout << "Select an option:\n"

                  << "1. Caesar Encrypt\n"

                  << "2. Caesar Decrypt\n"

                  << "3. Affine Encrypt\n"

                  << "4. Affine Decrypt\n"

                  << "5. Exit\n";

**int** choice;

        std::cout << "Enter your choice (1-5): ";

        std::cin >> choice;

**if** (choice == 5) {

            // 4. Exit the program

**break**;

        }

**const** **char**\* text;

**const** **char**\* result;

**switch** (choice) {

**case** 1:

                  // Caesar Encrypt

                  {

                      // 使用字符数组接收用户输入

**char** inputText[256];

                      std::cout << "Enter the text to encrypt: ";

                      std::cin >> inputText;

**int** caesarOffset;

                      std::cout << "Enter the Caesar offset: ";

                      std::cin >> caesarOffset;

                      // 将字符数组转换为 const char\*

                      text = strdup(inputText);

                      result = caesarEncrypt(text, caesarOffset);

                      std::cout << "Caesar Encrypted: " << result << std::endl;

                      // 释放动态分配的内存

                      free(**const\_cast**<**char**\*>(text));

                  }

**break**;

**case** 2:

                  // Caesar Decrypt

                  {

**char** inputText[256];

                      std::cout << "Enter the text to decrypt: ";

                      std::cin >> inputText;

**int** caesarDecryptOffset;

                      std::cout << "Enter the Caesar offset for decryption: ";

                      std::cin >> caesarDecryptOffset;

                      text = strdup(inputText);

                      result = caesarDecrypt(text, caesarDecryptOffset);

                      std::cout << "Caesar Decrypted: " << result << std::endl;

                      free(**const\_cast**<**char**\*>(text));

                  }

**break**;

**case** 3:

                  // Affine Encrypt

                  {

**char** inputText[256];

                      std::cout << "Enter the text to encrypt: ";

                      std::cin >> inputText;

**int** affineA, affineB;

                      std::cout << "Enter the Affine parameters (a b): ";

                      std::cin >> affineA >> affineB;

                      text = strdup(inputText);

                      result = affineEncrypt(text, affineA, affineB);

                      std::cout << "Affine Encrypted: " << result << std::endl;

                      free(**const\_cast**<**char**\*>(text));

                  }

**break**;

**case** 4:

                  // Affine Decrypt

                  {

**char** inputText[256];

                      std::cout << "Enter the text to decrypt: ";

                      std::cin >> inputText;

**int** affineDecryptA, affineDecryptB;

                      std::cout << "Enter the Affine parameters (a b) for decryption: ";

                      std::cin >> affineDecryptA >> affineDecryptB;

                      text = strdup(inputText);

                      result = affineDecrypt(text, affineDecryptA, affineDecryptB);

                      std::cout << "Affine Decrypted: " << result << std::endl;

                      free(**const\_cast**<**char**\*>(text));

                  }

**break**;

**default**:

                std::cout << "Invalid choice. Please enter a number between 1 and 5." << std::endl;

        }

    }

    // 5. Unload the dynamic library

    dlclose(libHandle);

**return** 0;

}

Тестирование кода

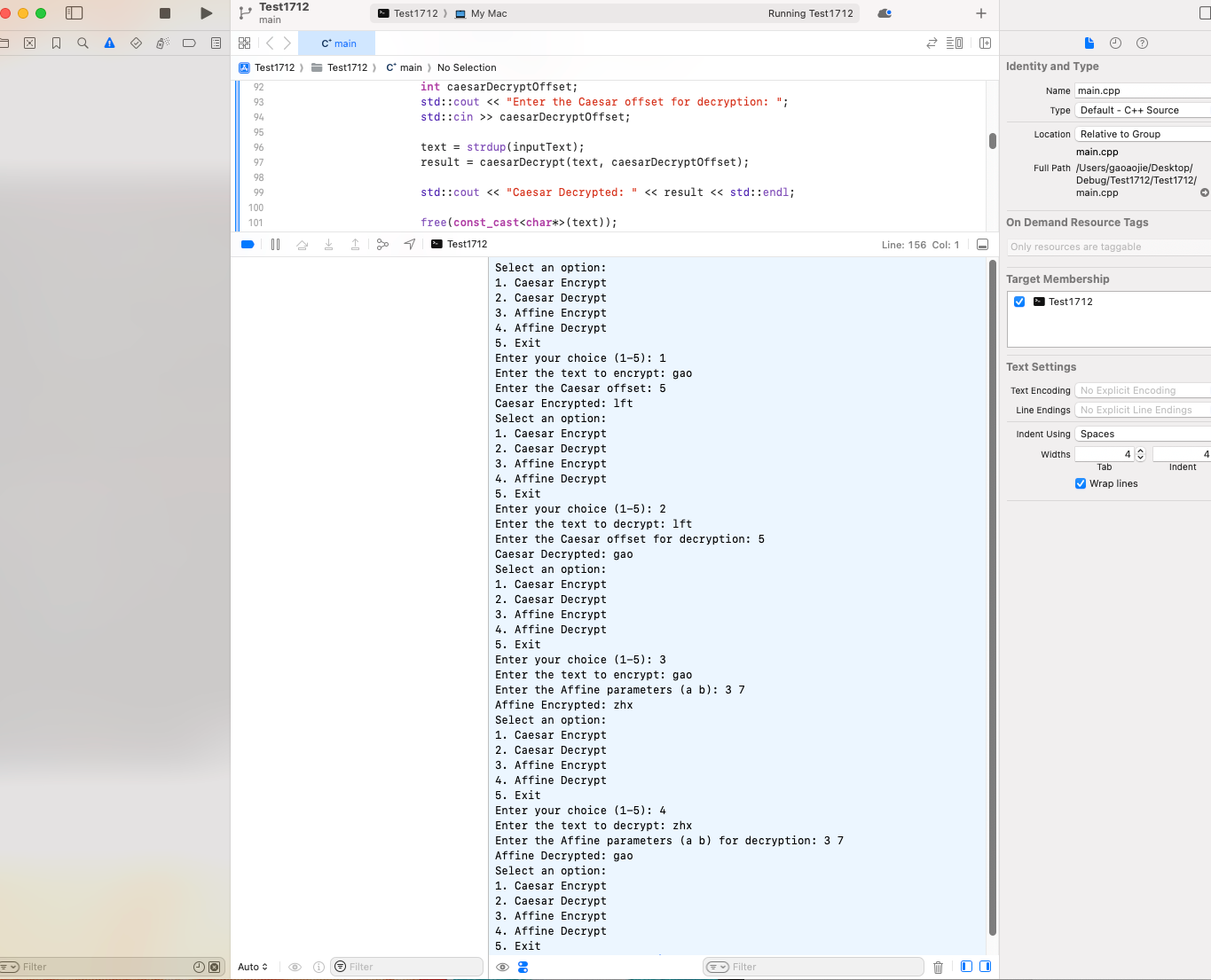


Рисунок 1 – Результат теста кода

На рисунке показан процесс шифрования и дешифрования "гао" с помощью шифров Цезаря и аффинных шифров.

## Вывод

В этом эксперименте были изучены классические криптографические алгоритмы и выбраны шифр Цезаря и аффинный шифр для написания DLL и тестирования DLL. На предыдущем курсе мы научились писать DLL в среде windows с помощью visualstudio, а в этот раз написали в среде MACos с помощью программы Xcode.