Магамедова Асет Зайнаевна
старший преподаватель
ФГБОУ ВО «Чеченский государственный
педагогический университет»
г. Грозный, Чеченская Республика

ИЗУЧЕНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ПРИ РАБОТЕ С ФАЙЛАМИ НА ПРИМЕРАХ РАЗРАБОТКИ ПРИЛОЖЕНИЙ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ ТЕКСТА

Аннотация*: изучение программирования предполагает работу с файлами, с текстовыми и бинарными. Основываясь на собственном опыте, автор выдвигает предположение, что любая практическая работа выполняется быстрее если создан четкий образ того конечного продукта, который должен быть получен в результате затраченного времени и труда. При этом процесс работы должен быть разбит на участки, которые усложняются по мере прохождения пути. Данная статья представляет варианты алгоритмов, которые осуществляют метод замены символов при работе с файлами.*

Ключевые слова*: программирование, C++, шифрование, дешифрование, информационная безопасность, текстовые файлы.*

Между такими дисциплинами как программирование и информационная безопасность существует тесная связь и целью данной статьи является отражение данной связи при разборе темы «Программирование с файлами на языке C++» на примере разработки приложений шифрования путем простой замены шрифта и дешифрования, которая преобразуют текст в исходное состояние.

Шифрование и дешифрование относятся к криптографическим методам. Криптографические методы обеспечивают безопасность в таких направлениях как, передача/хранение данных; аутентификация; генерация электронной цифровой подписи и другие. Шифрование – это обратимое преобразование данных с целью их сокрытия от посторонних. Методов шифрования было придумано множество – от шифров простой замены до принципиально сложных, использующих

двоичное сложение и т.д. Почти все методы шифрования применяют ключ шифрования. Для замены символов исходного текста применяются самые различные варианты других символов: буквы алфавита, различные знаки и т.п.

Есть несколько способов работы с файлами с использованием языков C и C++. Самый распространенный связан со структурой FILE. Программирование с использованием текстового файла предполагает при его открытии связывание с потоком ввода-вывода. Выводимая информация записывается в поток, вводимая информация считывается из потока. Когда поток открывается для ввода-вывода, он связывается со стандартной структурой типа FILE. Открытие файла осуществляется с помощью функции fopen(), возвращающей указатель на FILE: «FILE *fopen (name, type)», где name – имя открываемого файла (включая путь), type – указатель на строку символов, определяющих способ доступа к файлу.

Доступ предполагает открытия файла для чтения, записи, добавления записи и т.д. В частности, значение «r» для переменной type, означает, что документ открывают для чтения (при этом он должен существовать), «w» — для записи. Если при открытия файла обнаружена ошибка, то возвращается значение NULL. Функция fclose() закрывает поток или потоки, связанные с открытыми при помощи функции fopen() файлами.

Для чтение символа из файла используют функцию char fgetc(поток). Аргументом функции является указатель на поток типа FILE. Функция возвращает код считанного символа. Если достигнут конец файла или возникла ошибка, возвращается константа EOF. Для записи символа в файл применяют функция двух переменных: fputc(символ, поток). Аргументами функции являются символ и указатель на поток типа FILE. Функция возвращает код считанного символа.

Методы fscanf() и fprintf() аналогичны scanf() и printf(), но работают с файлами данных, и имеют первый аргумент – указатель на файл: fscanf (поток, «ФорматВвода», аргументы), fprintf (поток, «ФорматВывода», аргументы).

При работе со строками также можно воспользоваться функциями fgets() и fputs(): fgets (УказательНаСтроку, КоличествоСимволов, поток), fputs (казательНаСтроку, поток).

Рассмотрим приложение, которое создает два текстовых файла, один для записи и считывания, другой – только записи. Предположим, что в самой программе задан некоторый текст (предложение). Алгоритм записывает информацию в первый файл. Для второго файла – приложение считывает текст из первого, шифрует и записывает во второй файл. Шифрование осуществляется только для имеющихся в исходном предложении символов.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
const int N=100;
int main() {
setlocale (LC_ALL,"rus");
FILE *S1, *S2;
char x[N], y[N];
S1 = fopen("ST1.txt", "w");
fprintf(S1, "Сегодня пасмурная погода");
fclose(S1);
S1 = fopen("ST1.txt", "r");
S2 = fopen("ST2.txt", "w");
fgets(y, 100, S1);
printf("%s", y);
for(int i=0; i<strlen(y); i++)
{
    if(y[i]=='a') y[i]='!';
    if(y[i]=='e') y[i]='@';
    if(y[i]=='c') y[i]='#';
    if(y[i]=='C') y[i]='+';
    if(y[i]=='п') y[i]='*';
```

```
        if(y[i]=='н') y[i]=':';
        if(y[i]=='м') y[i]='?';
        if(y[i]=='д') y[i]='$';
        if(y[i]=='я') y[i]='&';
        if(y[i]=='у') y[i]='№';
        if(y[i]=='г') y[i]='9';
        if(y[i]=='о') y[i]='%';
        if(y[i]=='р') y[i]='>';
    }
    fclose(S1);
    fprintf(S2, "%s\n", y);
    fclose(S2);
    return 0;
    }
```

Усложним задачу, составим приложение и для дешифрования. В итоге, должно быть два приложения, один для шифрования, другой – дешифрования. Для работы создадим три текстовых файла, первый из которых содержит исходный текст для шифрования.

Запись в файл при выполнении данного задания выполняется немного иначе, чем в первом варианте. Для работы с файлами используем заголовочный файл <fstream>. В нем определены несколько классов и подключены заголовочные файлы <ifstream> и <ofstream>. С использованием потока ifstream осуществляется считывание из файла, запись в файл происходит при использовании потока ofstream.

В языке программирования C++ стандартная библиотека ввода-вывода организована как иерархия шаблонов классов. В классах ios_base и ios определены методы, которые являются общими для входных и выходных потоков. Класс ios в числе множества других поддерживает методы clear (сбрасывает управляющие состояния) и eof (проверяет на конец файла). Кроме того, класс ios, в числе

множества других, поддерживает следующие режимы работы потока: in (входной файл) и out (выходной файл).

Класс ostream, обеспечивает методами для записи данных в буфер потока. Перечислим некоторые из них, которые необходимы для работы: объект cout (соответствующий стандартному выводу), объект cerr (соответствующий стандартному выводу для ошибок), метод put (выводит символ), seekp (устанавливает позицию для функции put), tellp (читает позицию указателя для функции put), write (пишет последовательность байтов) и т. д. От класса ostream наследуются класс ofstream, который содержит дополнительные методы: open (открыть файл), close (закрыть файл)

Посимвольное считывание многострочного текстового файла в вектор строк включает применение STL (Стандартная Библиотека Шаблонов). STL предоставляет набор хорошо сконструированных и согласованно работающих вместе обобщённых компонентов C++. Библиотека содержит пять основных видов компонентов, из которых двое используются в данной работе: контейнер (управляет набором объектов в памяти), итератор (средство доступа к одержимому контейнера). Например, функции begin() и end() – функции-члены контейнеров, которые возвращают правильные типы итераторов. В программной строке, которая содержит данные функции: copy(v.begin(), v.end(), ostream_iterator<string>(cout)), метод copy перемещает строки из стандартного ввода в вектор, но так как вектор предварительно не размещён в памяти, используется итератор вставки, чтобы вставить в вектор строки одну за другой. В итоге, copy копирует вектор в поток *cout* и данные выводятся в консоль.

Сохранение зашифрованного файла без потери форматирования означает, что сохраняются все переносы строк. То есть предполагаем использование векторов таким образом, чтобы каждый элемент вектора является одним абзацем.

После открытия текстового файла для чтения, посимвольно считываем из него символы в файловый поток in: c=in.get(). Имя потока и переменной одновременной – in, но можно задать и другое имя, главное, не нарушать правила объявления переменных. В скобках прописана константа, обозначающая путь к

файлу (FName). EOF — это символ конца файла, который каждый раз проверяет не является ли считываемый символ концом файла. Внутри оператора цикла идет сборка строки, проверка с на то, что он не является символом переноса строки и, если он является переносом, то собранная строка записывается в вектор. Вектор в C++ — это замена стандартному динамическому массиву, память для которого выделяется вручную, с помощью оператора new. Например, вектор состоящий 5 целых элементов имеет вид: vector<int> v(5), а запись vector<string>v – означает вектор, состоящий из строк. Для добавления нового элемента в конец вектора используется метод push_back(). Кроме push_back() при работе с векторами используют методы: pop_back(), для удаления последнего элемента; clear(), для удаления всех элементов; empty(), для проверки вектора на пустоту. После записи строки в вектор, строка обязательно очищается, иначе данная строка продолжит собираться с уже накопленными в ней символами. После выполнения цикла дописывается последняя строка в вектор строк.

В разработке приложений применяют методы, описание которых представлены в данной работе.

```
//Шифрование
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include <iterator>
using namespace std;
int main(){
int c=0; string str;
vector<string>v;
const char *F1="Text_.txt", *F2="TextN.txt";
ifstream in(F1);
while((c = in.get()) != EOF)
{
```

```
        if (char(c)!='\n')
        switch(char(c))
        {
                case 'A': { char(c)= 'f'; str+=char(c); break;  }
                …
        }
        else {v.push_back(str);  str.clear(); }
}
v.push_back(str);
in.close();
copy(v.begin(),v.end(),ostream_iterator<string>(cout,"\n\n"));
ofstream fout(F2);
copy(v.begin(),v.end(),ostream_iterator<string>(fout, "\n"));
fout.close();
v.clear();
return 0;
}
```

//Дешифрование

```
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include <iterator>
using namespace std;
int main()
{
int c=0;  string str;
vector<string> v;
const char *F1="TextN.txt", *F2="Text0.txt";
ifstream in(F1);
```

```
while((c = in.get()) != EOF)
{
    if (char(c)!='\n')
    switch(char(c))
    {
        case 'f': { char(c)= 'A'; str+=char(c); break; }
        …
    }
    else {  v.push_back(str); str.clear();   }
}
v.push_back(str);
in.close();
copy(v.begin(),v.end(),ostream_iterator<string>(cout,"\n\n"));
ofstream fout(F2);
copy(v.begin(),v.end(),ostream_iterator<string>(fout, "\n"));
fout.close();
v.clear();
return 0;
}
```

Таким образом, составлены две программы: первая программа считывает исходный текст с файла «Text_.txt», производит замену символов, осуществляет запись шифрованного текста в файл «TextN.txt» и выводит его в консоль (при этом предполагается сохранение форматирования), вторая программа считывает информацию из документа «TextN.txt», производит операцию дешифрования, осуществляет запись текста в файл «Text0.txt» и выводит его в консоль (эту команду можно удалить). То есть, в результате получен исходный текст.

Результаты работы возможно будут актуальны в следующих областях: программирование, информационная безопасность, кодирование данных.

Список литературы

1. Лафоре Р. Объектно-ориентированное программирование в С++. – СПб., 2004. – 922 с.

2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб.: Питер, 2012. – 464 с.

3. Павловская Т.А. С/С++. Структурное программирование: Практикум / Т.А. Павловская, Ю.А. Щупак. – СПб.: Питер, 2005. – 239 с.

4. Побегайло А.П. С/С++ для студента. – СПб.: БХВ-Петербург, 2006. – 528 с.

5. Стивен П. Язык программирования С++. Лекции и упражнения / Пер. с англ. – 5-е изд. – М., 2007. – 1184 с

6. Страуструп Б. Язык программирования С++. Специальное издание – М.: Бином-Пресс, 2008. – 1104 с.

7. Работа с файлами в языке Си [Электронный ресурс]. – Режим доступа: https://studlib.info/informatika/134107-rabota-s-faylami-v-yazyke-si/

8. [Электронный ресурс]. – Режим доступа: http://ci-plus-plus-snachala.ru/