C++ 基础与深度解析
第八章作业 – 情报督察 思路提示
2022-06-20

主讲人 天哲

纲要

- 加密/解密
- 程序设计
- 码本检查
- 字母表密码本
- 扩展1
- 常见问题
- 其他

加密/解密

深蓝学院 shenlanxueyuan.com

- 在本作业中，加密/解密的本质是替换
- 加密/解密的操作是对称的
- 关键在于构建映射
- 可以使用std::map 或者std::vector来创建映射表，这里以vector为例

示例

```
std::vector<uint8_t> anyfile; //文件
std::vector<uint8_t> encryptbook(26); //密码本
std::vector<uint8_t> decrybook(26); //解码本
char c, x;//c代表处理前的字符，x代表处理后的字符
```

示例

```
original   ="abcdefghijklmnopqrstuvwxyz"
encryptbook="qwertyuiopasdfghjklzxcvbnm"
decryptbook="kxvmcnophqrszyijadlegwbuft"
encryptbook['k' - 'a'] = 'a';
decryptbook['a' - 'a'] = 'k';
```

程序的设计

1. 读入码本文件，初步筛查，检查是否有非法字符。对于字母码表，可以统一变成大写/小写方便下面的处理。对于数字码表，可以检查有没有超过255的数，并将结果储存到vector<uint8_t>或者数组中
2. 进一步检查码表是否完备，是否包含了所有需要的元素 虽然批改作业的时候，对于码本检查不一定做强制要求，但是经常检查参数错误可以尽早定位问题，方便以后debug
3. 根据密码表，生成所需的解码表
4. 读入需要加密的内容
5. 使用密码/解码表进行相对应的替换，实现加密/解密功能

码本的检查

深蓝学院 shenlanxueyuan.com

- 不论是对于字母码表，还是数字码表，都有：范围内的uint8_t 或者char 都出现且仅出现1次

- 循环检查,也可以使用std::sort，因为上面的结论，所以对于任意的码表，经过排序后一定会等于abcdefg..或者0,1,2...255

```
//全部转换为小写
std::transform(str.begin(),
str.end(), str.begin(),
[](unsigned char c)
{ return std::tolower(c); });
```

```
bool codeBookIsValid(const std::string &input)
{   constexpr size_t alphabetSize = 26;
    if (input.size() != alphabetSize)
                return false;
    std::vector<bool> exists(alphabetSize, false);
    for (const auto &c : input)
    {
            // if alreay exists, means duplicate
            if (c<'a'||c>'z'||exists[c - 'a'])
                return false;
            exists[c - 'a'] = true;
     }
    return true;
}

bool codeBookIsValid(const std::string &input)
{
    std::string reference = "abcdefghijklmnopqrstuvwxyz";
    std::string buf = input;
    std::sort(buf.begin(), buf.end());
    return buf == reference;
}
```

字母表密码本

示例

```
original   ="abcdefghijklmnopqrstuvwxyz"
encryptbook="qwertyuiopasdfghjklzxcvbnm"
decryptbook="kxvmcnophqrszyijadlegwbuft"
encryptbook['k' - 'a'] = 'a';
decryptbook['a' - 'a'] = 'k';
```

- 加密：

对于任意一个a-z之间char c 和char x 有：

```
encryptbook[c-'a']=x;
```

加密程序

```
 for (char &c : anyfile)
{
    if (c >= 'a' && c <= 'z')
        c = encryptbook[c - 'a'];
    //... if (c>='A'&&c<='Z)
}
```

- 解密：

对于解码本，我们希望有：

```
decyptbook[x-'a']=c;
decryptbook[encryptbook[c-'a']-'a']=c
```

生成解码本

```
std::vector<uint8_t> decryptbook (26);
for (const auto &c : encryptbook)
        decryptbook[encryptbook[c - 'a'] - 'a'] = c;
```

解密程序

```
 for (char &c : anyfile)
{
    if (c >= 'a' && c <= 'z')
        c = decryptbook [c - 'a'];
    //... if (c>='A'&&c<='Z)
}
```

注意到加密/解密的形式是对称的，仅仅是把encryptbook和decryptbook的差异

扩展1

深蓝学院 shenlanxueyuan.com

- 加密：

任何一个数据可以按二进制字节表示，这里扩展1,2的意思就是把数据当作二进制数据流处理。所以可以适用于任何形式的文件

对于任意一个字节(或者说uint8_t, char 一样都占8位可以保存0-255) i, 有：

```
encryptbook[i] = x;
```

加密程序

```
 for (uint8_t &i : anyfile)
{
    i = encryptbook[i];
}
```

示例：’ \’ 前缀代表它是1个uint8_t

```
{\0,\2,\3} --> {\1,\3,\4}
encryptbook={\1,\2,\3, ... \255,\0}
decryptbook={\255,\0,\1,\2,\3,\4 ....\254}
encryptbook[2] = 3;
decryptbook[3] = 2;
```

- 解密：

对于解码本，我们希望有：

```
decyptbook[x]=i;
decryptbook[encryptbook[i]]=i
```

生成解码本

```
std::vector<uint8_t> decryptbook (256);
for (const auto i : encryptbook)
        decryptbook[encryptbook[i]] = i;
```

解密程序

```
 for (uint8_t &i : anyfile)
{
    i = decryptbook[i];
}
```

注意到加密/解密的形式是对称的，仅仅是把encryptbook和decryptbook的差异

扩展1

深蓝学院 shenlanxueyuan.com

- 扩展1的通用性更强，对称性更明显
- 代码更加简单

可以用扩展一的方法来解决字母表密码本：
encryptbook={\0,\1,\2,...,'Q','W','E',.....'q','w','e' ... \255};

可以进一步抽象，使用一个encode函数，dict可以是密码/解码表，这样一个函数就解决了加密/解密

```
void encode(std::vector<uint8_t> &vec,
            const std::vector<uint8_t> &dict)
{
    if (vec.empty() || dict.empty())
        return;
    for (auto &i : vec)
        i = dict[i];
    return;
}
```

```
//input for example, real codebook read from file
std::string input = "qwertyuiopasdfghjklzxcvbnm";
const uint8_t interval = 'a' - 'A';
for (uint16_t i = 0; i < encryptbook.size(); ++i)
{
    if (i >= 'a' && i <= 'z')
    {
        encryptbook[i] = input[i - 'a'];
        //also do it for uppercase chars
        encryptbook[i - interval] = encryptbook[i] -
                                    interval;
    }
    else if (i >= 'A' && i <= 'Z')
    {
        //do nothing
    }
    else // encryptbook[0] =0
        encryptbook[i] = i;
}
//make decrypt book
for (const auto i : encryptbook)
{
    decryptbook[encryptbook[i]] = i;
}
```

常见问题

深蓝学院
shenlanxueyuan.com

1. 没有使用命令行参数，上一章作业应该学习过了，这次必须使用命令行参数，可以参考第二次作业补充说明
2. 如果参考其他人或者网上的代码，务必自己调试运行一下
3. 如果要添加自己的内容，必须往兼容性更广的添加：

→ 根据码本内容，自动判断加密的形式 ✓

→ 兼容大写的码本 ✓

→ 限定码本的文件名/后缀名，限定码本存放的文件夹 X

文件读写和其他


```cpp
std::vector<uint8_t> data_bytes;
std::ifstream ifs(inputFilePath, std::ios::in | std::ios::binary);
if (!ifs)
{
    std::cerr<<"Read Input File fails\n";
}
data_bytes.assign((std::istreambuf_iterator<char>(ifs)), std::istreambuf_iterator<char>());
ifs.close();

std::ofstream ofs(outputFilePath, std::ios::out | std::ios::binary);
if (!ofs)
{
    std::cerr<<"Open Output File fails\n";
}
ofs.write(reinterpret_cast<const char*>(data_bytes.data()), data_bytes.size());
ofs.close();
```

- Cpp本身就有很多STL算法，除了之前提到的std::sort外，例如std::shuffle, std::reverse 等都很实用，shuffle就可以拿来生成乱序的码表用
- std::string 和std::vector<uint8_t>是可以互相转化的

```cpp
std::vector<uint8_t> vec(str.begin(), str.end());
std::string str(vec.begin(), vec.end());
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening