

# 密码学lab1

1813540 陈鸿运

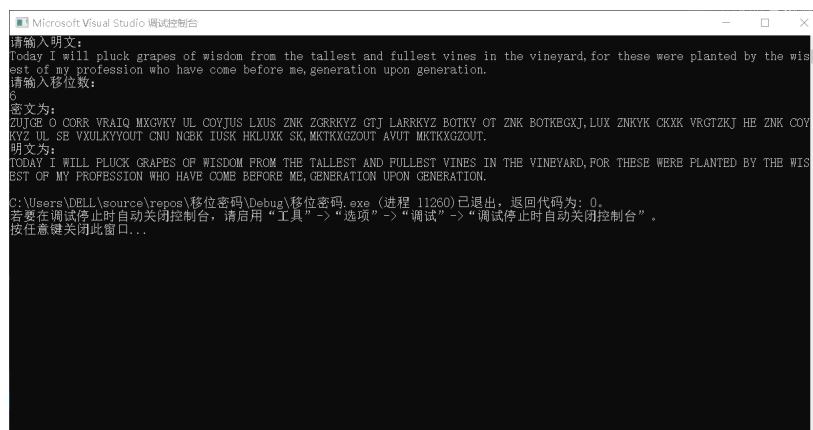
2020 年 11 月 21 日

## 1 移位密码算法

实现该算法的思路是对每个输入的字符进行判断，如果是小写字母则进行转化，利用ASCII码转化为大写字母。若是其他字符(如空格)则忽略。随后再次进行判断，取得每个大写字母的ASCII码之后转化为其在26个字母中对应的序号(从0-25)，利用密钥进行取模加法，最后再转化为相应的ASCII码。

对应的解密代码的思路与之类似，值得注意的是，解密算法中需要减去密钥值。但这有可能造成字母的ASCII码不正确，因此可以考虑将减密钥值的操作转化为加上26-密钥值。

算法运行的效果如图??：



```
Microsoft Visual Studio 调试控制台
请输入明文:
Today I will pluck grapes of wisdom from the tallest and fullest vines in the vineyard, for these were planted by the wisest of my profession who have come before me, generation upon generation.
请输入移位数:
6
密文为:
ZUJGE O CORR VRAIQ MXGVKY UL COYJUS LXUS ZNK ZGRKYZ GTJ LARRKYZ BOTKY OT ZNK BOTKEGXJ, LUX ZNKYK CKXK VRGTZKJ HE ZNK COYKYZ UL SE VXULKYYOUT CNU NGBK IUSK HKLUXK SK, MKTKXGZOUT AVUT MKTKXGZOUT.
明文为:
TODAY I WILL PLUCK GRAPES OF WISDOM FROM THE TALLEST AND FULLEST VINES IN THE VINEYARD, FOR THESE WERE PLANTED BY THE WISEST OF MY PROFESSION WHO HAVE COME BEFORE ME, GENERATION UPON GENERATION.
C:\Users\DELL\source\repos\移位密码\Debug\移位密码.exe (进程 11260) 已退出，返回代码为: 0。
若要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

图 1: 移位密码算法程序运行效果

其明文、密文与解密后的密文如下，其中密钥为6：

明文：

Today I will pluck grapes of wisdom from the tallest and fullest vines in the vineyard, for these were planted by the wisest of my profession who have come before me, generation upon generation.

密文：

ZUJGE O CORR VRAIQ MXGVKY UL COYJUS LXUS ZNK ZGRKYZ GTJ LARRKYZ BOTKY OT ZNK BOTKEGXJ, LUX ZNKYK CKXK VRGTZKJ HE ZNK COYKYZ UL SE VXULKYYOUT CNU NGBK IUSK HKLUXK SK, MKTKXGZOUT AVUT MKTKXGZOUT.

解密密文：

TODAY I WILL PLUCK GRAPES OF WISDOM FROM THE TALLEST  
AND FULLEST VINES IN THE VINEYARD, FOR THESE WERE PLANTED  
BY THE WISEST OF MY PROFESSION WHO HAVE COME BEFORE  
ME, GENERATION UPON GENERATION.

可以看到，程序正确地进行了加密与解密操作。

## 2 破解移位密码算法

移位密码算法的破解非常简单，直接暴力对每个密钥进行遍历得到明文，之后分析其是否正确即可。

下面是同学给我的密文：

JYFWAVSVNF, AOL ALJOUVSVNF VM THRPUN HUK IYLHRPUN  
JVKLZ HUK JPWOLYZ, OHZ MBYUPZOLK HTLYPJH DPAO LEJLSS-  
LUA WYVALJAPVU MVY PAZ AYHUZTPAALK KVJBTLUAZ HUK  
DPAO PAZ ILZA PUALSSPNLUJL.

下面我们尝试利用程序破解该密文，输入密文：

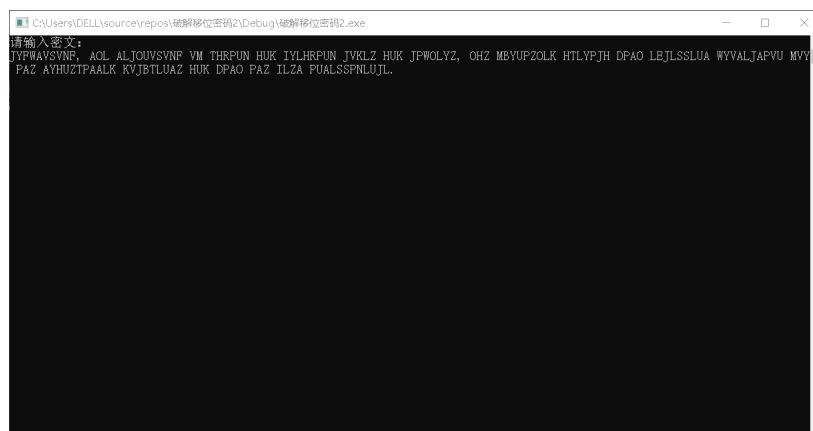


图 2: 密文输入

运行程序，并得到结果：

key=1时明文为：

IXEVZURUME, ZNK ZKINTURUME UL SGQOTM GTJ HXKGQOTM  
IUJKY GTJ IOVNKXY, NGY LAXTOYNKJ GSKXOIG COZN KDIKR-

RKTZ VXUZKIZOUT LUX OZY ZXGTYSOZZKJ JUIASKTZY GTJ COZN  
OZY HKYZ OTZKRROMKTIK.

key=2时明文为:

HWDUYTQTLD, YMJ YJHMSTQTLD TK RFPNSL FSI GWJF-  
PNSL HTIJX FSI HNUMJWX, MFX KZWSNXMJI FRJWNHF BNYM  
JCHJQQJSY UWTYJHYNTS KTW NYX YWFSXRNYJI ITHZRJSYX  
FSI BNYM NYX GJXY NSYJQQNLJSHJ.

key=3时明文为:

GVCTXSPSKC, XLI XIGLRSPSKC SJ QEOMRK ERH FVIEOMRK  
GSHIW ERH GMTLIVW, LEW JYVRMWLIH EQIVMGE AMXL IBGIP-  
PIRX TVSXIGXMSR JSV MXW XVERWQMXXIH HSGYQIRXW ERH  
AMXL MXW FIWX MRXIPPMKIRGI.

key=4时明文为:

FUBSWRORJB, WKH WHFKQRORJB RI PDNLQJ DQG EUHDNLQJ  
FRGHV DQG FLSKHUV, KDV IXUQLVKHG DPHULFD ZLWK HAFHOOHQW  
SURWHFWLRQ IRU LWV WUDQVPLWWHG GRFXPHQWV DQG ZLWK  
LWV EHVW LQWHOOLJHQFH.

key=5时明文为:

ETARVQNQIA, VJG VGEJPQNQIA QH OCMKPI CPF DTGCMKPI  
EQFGU CPF EKRJGTU, JCU HWTPKUJGF COGTKEC YKVJ GZEG-  
NNGPV RTQVGEVKQP HQT KVV VTCPUOKVVGF FQEWOGPVU  
CPF YKVJ KVV DGUV KPVGNKIGPEG.

key=6时明文为:

DSZQUPMPHZ, UIF UFDIOPMPHZ PG NBLJOH BOE CSFBLJOH  
DPEFT BOE DJQIFST, IBT GVSOJTIFE BNFSJDB XJUI FYDFMM-  
FOU QSPUFDUJPO GPS JUT USBOTNJUUFE EPDVNFOUT BOE XJUI  
JUT CFTU JOUFMMJHFODF.

key=7时明文为:

CRYPTOLOGY, THE TECHNOLOGY OF MAKING AND BREAK-  
ING CODES AND CIPHERS, HAS FURNISHED AMERICA WITH EX-  
CELLENT PROTECTION FOR ITS TRANSMITTED DOCUMENTS AND  
WITH ITS BEST INTELLIGENCE.

key=8时明文为:

BQXOSNKNFX, SGD SDBGMNKNFX NE LZJHMF ZMC AQDZJHMF  
BNCDR ZMC BHOGDQR, GZR ETQMHRGDC ZLDQHBZ VHSG DWB-  
DKKDMS OQNSDBSHNM ENQ HSR SQZMRLHSSDC CNBTLDMSR  
ZMC VHSG HSR ADRS HMSDKKHFDMBD.

key=9时明文为:

APWNRMJMEW, RFC RCAFLMJMEW MD KYIGLE YLB ZPCYIGLE  
AMBCQ YLB AGNFCPQ, FYQ DSPLGQFCB YKCPGAY UGRF CVACJJ-  
CLR NPMRCARGML DMP GRQ RPYLQKGRRCB BMASKCLRQ YLB  
UGRF GRQ ZCQR GLRCJJGECLAC.

key=10时明文为:

ZOVMLLILDV, QEB QBZEKLILDV LC JXHFKD XKA YOBXH-  
FKD ZLABP XKA ZFMEBOP, EXP CROKFPEBA XJBOFZX TFQE  
BUZBIIBKQ MOLQBZQFLK CLO FQP QOXKPJFQQBA ALZRJBKQP  
XKA TFQE FQP YBPQ FKQBIIFFDBKZB.

key=11时明文为:

YNULPKHKCU, PDA PAYDJKHKCU KB IWGEJC WJZ XNAWGEJC  
YKZAO WJZ YELDANO, DWO BQNJEODAZ WIANEYW SEPD ATYAH-  
HAJP LNKPAYPEKJ BKN EPO PNWJOIEPPAZ ZKYQIAJPO WJZ SEPD  
EPO XAOP EJPAHHECAJYA.

key=12时明文为:

XMTKOJGJBT, OCZ OZXCIJGJBT JA HVFDIB VIY WMZVFDIB  
XJYZN VIY XDKCZMN, CVN APMIDNCZY VHZMDXV RDOC ZSXZG-  
GZIO KMJOZXODJI AJM DON OMVINHD00ZY YJXPHZION VIY  
RDOC DON WZNO DIOZGGDBZIXZ.

key=13时明文为:

WLSJNIFIAS, NBY NYWBHIFIAS IZ GUECHA UHX VLYUECHA  
WIXYM UHX WCJBYLM, BUM ZOLHCMBYX UGYLCWU QCNB YR-  
WYFFYHN JLINYWNCIH ZIL CNM NLUHMGCNNYX XIWOGYHNM  
UHX QCNB CNM VYMN CHNYFFCAYHWY.

key=14时明文为:

VKRIMHEHZR, MAX MXVAGHEHZR HY FTDBGZ TGW UKXTD-  
BGZ VHWXL TGW VBIAXKL, ATL YNKGBLAXW TFXKBVT PBMA  
XQVXEEXGM IKHMXVMBHG YHK BML MKTGLFBMMXW WHVN-

FXGML TGW PBMA BML UXLM BGMXEEDBZXGVX.

key=15时明文为:

UJQHLGDGYQ, LZW LWUZFGDGYQ GX ESCAFY SFV TJWSCAFY UGVWK SFV UAHZWJK, ZSK XMJFAKZWV SEWJAUS OALZ WPUWDDWFL HJGLWULAGF XGJ ALK LJSFKEALLWV VGUMEWFLK SFV OALZ ALK TWKL AFLWDDAYWFUW.

key=16时明文为:

TIPGKFCFXP, KYV KVTYEFKFXP FW DRBZEX REU SIVRBZEX TFUVJ REU TZGYVIJ, YRJ WLIEZJYVU RDVIZTR NZKY VOTVC-CVEK GIFKVTKZFE WFI ZKJ KIREJDZKKVU UFTLDVEKJ REU NZKY ZKJ SVJK ZEKVCCZXVETV.

key=17时明文为:

SHOFJEBEWO, JXU JUSXDEBEWO EV CQAYDW QDT RHUQAYDW SETUI QDT SYFXUHI, XQI VKHDYIXUT QCUHYSQ MYJX UNSUBUDJ FHEJUSJYED VEH YJI JHQDICYJJUT TESKCUDJI QDT MYJX YJI RUIJ YDJUBBYWUDSU.

key=18时明文为:

RGNEIDADVN, IWT ITRWCDADVNDU BPZXCV PCS QGTPZXCV RDSTH PCS RXEWTGH, WPH UJGCXHWTS PBTGXRP LXIW TMRTAATCI EGDITRIXDC UDG XIH IGPCHBXIITS SDRJBTCIH PCS LXIW XIH QTHI XCITAAXVTCRT.

key=19时明文为:

QFMDHCZCUM, HVS HSQVBCZCUM CT AOYWBU OBR PFSOYWBU QCRSG OBR QWDVSFG, VOG TIFBWGVSR OASFWQO KWHV SLQSZZSBH DFCHSQHWCB TCF WHG HFOBGAWHHSR RCQIASBHG OBR KWHV WHG PSGH WBHSZZWUSBQS.

key=20时明文为:

PELCGBYBTL, GUR GRPUABYBTL BS ZNXVAT NAQ OERNXVAT PBQRF NAQ PVCUREF, UNF SHEAVFURQ NZREVPN JVGURKPRYYRAG CEBGRPGVBA SBE VGF GENAFZVGGRQ QBPHZRAGF NAQ JVGU VGF ORFG VAGRYVTRAPR.

key=21时明文为:

ODKBFAXASK, FTQ FQOTZAXASK AR YMWUZS MZP NDQMWUZS

OAPQE MZP OUBTQDE, TME RGDZUETQP MYQDUOM IUFT QJO-  
QXXQZF BDAFQOFUAZ RAD UFE FDMZEYUFFQP PAOGYQZFE MZP  
IUFT UFE NQEF UZFQXXUSQZOQ.

key=22时明文为:

NCJAEZWZRJ, ESP EPNSYZWZRJ ZQ XLVTYR LYO MCPLV-  
TYR NZOPD LYO NTASPCD, SLD QFCYTDSPO LXPCTNL HTES PIN-  
PWWPYE ACZEPNETZY QZC TED ECLYDXTEEPO OZNFXPYED  
LYO HTES TED MPDE TYEPWWTRPYNP.

key=23时明文为:

MBIZDYVYQI, DRO DOMRXYVYQI YP WKUSXQ KXN LBOKUSXQ  
MYNOC KXN MSZROBC, RKC PEBXSCRON KWOBSMK GSDR OHMOVVOXD  
ZBYDOMDSYX PYB SDC DBKXCWSDDON NYMEWOXDC KXN GSDR  
SDC LOCD SXDOVVSQOXMO.

key=24时明文为:

LAHYCXUXPH, CQN CNLQWXUXPH XO VJTRWP JWM KAN-  
JTRWP LXMNB JWM LRYQNAB, QJB ODAWRBQNM JVNARLJ FRCQ  
NGLNUUNWC YAXCNLCRXW OXA RCB CAJWBVRCCNM MXLD-  
VNWCB JWM FRCQ RCB KNBC RWCNUURPNWLN.

key=25时明文为:

KZGXBWTWOG, BPM BMKPVWTWOG WN UISQVO IVL JZMISQVO  
KWLMA IVL KQXPMZA, PIA NCZVQAPML IUMZQKI EQBP MFKMTTMVB  
XZWBMBQWV NWZ QBA BZIVAUQBBML LWKCUMVBA IVL EQBP  
QBA JMAB QVBM TTQOMVKM.

经分析易得出, 密钥为7时, 得到的是有正确语义的字符串, 故密钥  
为7, 破译成功。

### 3 单表置换密码

实现该密码的思路是先使用vector容器存放进行置换的置换表, 第0项  
存放字母A转化的字符, 第二项存放字母B转化的字符, 以此类推。随后  
利用ASCII码进行操作, 思路与移位密码算法类似, 即先将字母都转为大  
写字母, 并将其对应的转化字母替换原来的字母, 这里我们将其存放到了  
ciphertext数组中。

算法运行效果如图??(这里我们选取实验文档中的例子作为密钥):

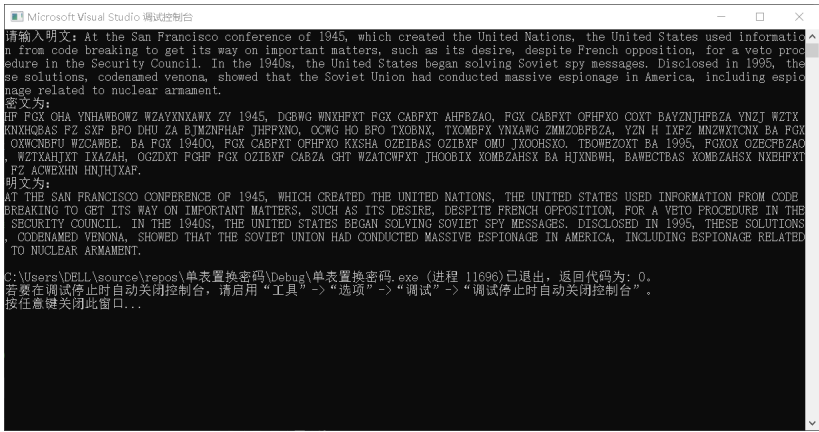


图 3: 单表置换密码算法程序运行效果

其明文、密文与解密后的密文如下:

明文:

At the San Francisco conference of 1945, which created the United Nations, the United States used information from code breaking to get its way on important matters, such as its desire, despite French opposition, for a veto procedure in the Security Council. In the 1940s, the United States began solving Soviet spy messages. Disclosed in 1995, these solutions, codenamed venona, showed that the Soviet Union had conducted massive espionage in America, including espionage related to nuclear armament.

密文:

HF FGX OHA YNHAWBOWZ WZAYXNXAWX ZY 1945, DGBWG  
WNXHFXXT FGX CABFXT AHFBZAO, FGX CABFXT OFHFXO COXT  
BAYZNJHFBZA YNZJ WZTX KNXHQBAS FZ SXF BFO DHU ZA BJMZNHFHAF  
JHFFXNO, OCWG HO BFO TXOBNX, TXOMBFX YNXAWG ZMM-  
ZOBFBZA, YZN H IXFZ MNZWXTCNX BA FGX OXWCNBFU WZ-  
CAWBE. BA FGX 1940O, FGX CABFXT OFHFXO KXSHA OZEIBAS  
OZIBXF OMU JXOOHSXO. TBOWEZOXT BA 1995, FGXOX OZECF-  
BZAO, WZTXAHJXT IXAZAH, OGZDXT FGHF FGX OZIBXF CABZA  
GHT WZATCWFXT JHOOBIX XOMBZAHSX BA HJXNBWH, BAWECT-



BAS XOMBZAHXS NXEHFXT FZ ACWEXHN HNJHJXAF.

解密密文：

AT THE SAN FRANCISCO CONFERENCE OF 1945, WHICH CREATED THE UNITED NATIONS, THE UNITED STATES USED INFORMATION FROM CODE BREAKING TO GET ITS WAY ON IMPORTANT MATTERS, SUCH AS ITS DESIRE, DESPITE FRENCH OPPOSITION, FOR A VETO PROCEDURE IN THE SECURITY COUNCIL. IN THE 1940S, THE UNITED STATES BEGAN SOLVING SOVIET SPY MESSAGES. DISCLOSED IN 1995, THESE SOLUTIONS, CODENAMED VENONA, SHOWED THAT THE SOVIET UNION HAD CONDUCTED MASSIVE ESPIONAGE IN AMERICA, INCLUDING ESPIONAGE RELATED TO NUCLEAR ARMAMENT.

可以看到，程序正确地进行了加密与解密操作。

## 4 破解密文

文档中给出的密文如下：

SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPN-  
BRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY  
QCNBR MF N XMRRJHAY JBRCGZPC GINBBCA JB RZGI N VNY  
SINS SIC MPJEJBNA QCCRNEC GNB MBAY HC PCGMTCPCD HY  
SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJXNBSR JB SIC SPNBRNGSJMB  
NPC NAJGC SIC MPJEJBNSMP MF SIC QCCRNEC HMH SIC PCGCJTCP  
NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRICR SM ENJB  
ZBNZSIMPJOC D GMBSPMA MF SIC QCCRNEC

我们首先对文本中的单个字母进行频率分析，得到如下结果：

10.6825	<i>C</i>
9.79228	<i>S</i>
9.19881	<i>N</i>
8.60534	<i>M</i>
8.30861	<i>J</i>
8.30861	<i>B</i>
6.82493	<i>P</i>
6.23145	<i>R</i>
5.34125	<i>I</i>
4.1543	<i>G</i>
3.56083	<i>X</i>
2.96736	<i>A</i>
2.67062	<i>E</i>
2.67062	<i>H</i>
2.37389	<i>Q</i>
2.07715	<i>F</i>
2.07715	<i>Y</i>
1.48368	<i>Z</i>
0.890208	<i>V</i>
0.890208	<i>D</i>
0.593472	<i>T</i>
0.296736	<i>O</i>
0	<i>K</i>
0	<i>L</i>
0	<i>U</i>
0	<i>W</i>

我们发现C与S的频率最高，由此我们猜测这二者极有可能为文本中字母频率最高的E和T。考虑到密文中频频出现的SIC，以及常用单词中占比最高的THE，几乎可以肯定，S对应于T，I对应于H，C对应于E。

我们在破译程序中输入这三个对应表项，运行程序，得到如下结果(见图??):

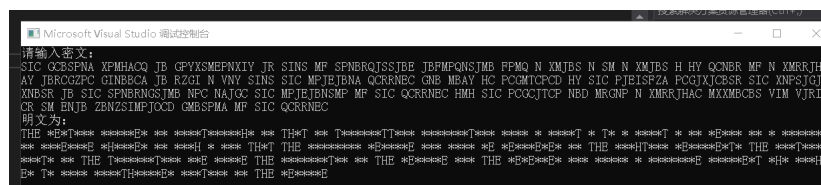


图 4: 已破译三个字母

随后我们考虑密文中出现较多的单个字母‘N’，在英语中，能够以单个字母出现多次的，除去某些代号，只有‘A’和‘I’有可能。但是能够在如此短的半句中出现这么多次，我们更倾向于考虑‘A’作为可能(图??):

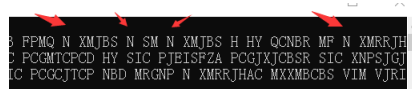


图 5: N出现得过于频繁

接下来我们再考虑常用单词的频率排在“THE”后的“TO”与“OF”。观察这两个单词，我们可以发现“TO”与“OF”之间存在一个共同字母‘O’，也就是说，我们可以根据这一点来推断。经过比较后，我们发现，密文中的“SM”与“MF”非常符合“TO”与“OF”之间的特性，即二者存在共同字母，并且‘S’我们之前已推出为‘T’，更加印证了我们的想法 (见图??):

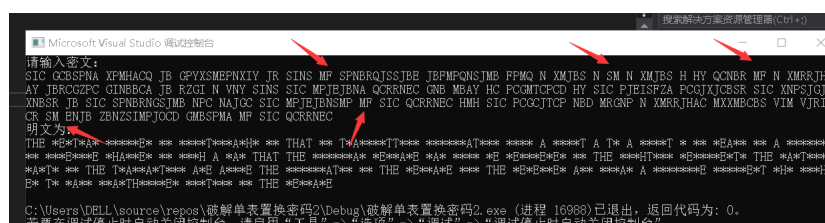


图 6: SM与MF十分可疑

由此推断‘M’为‘O’，‘F’就是‘F’，填入表中后运行程序得到结果：

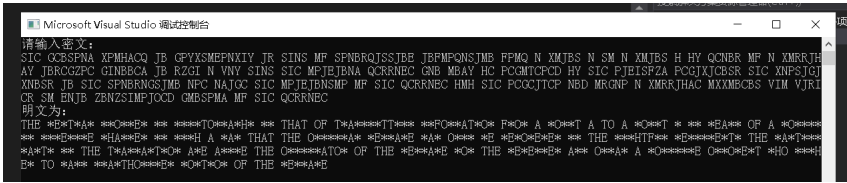


图 7: 已破译六个字母

随后我们又发现在密文中由‘J’组成的“JB”、“JR”也出现多次，字母‘A’已经推出，再考虑英语中重要的字母‘T’以及“IS”、“IT”、“IN”等常见单词，不妨推定，‘J’就是字母‘T’。

接下来我们就考虑“JB”与“JR”，由于之前的‘S’已推出为‘T’，那么剩下的两个应该就是‘S’与‘N’了，这里我们猜测 ‘B’为‘N’，‘R’为‘S’。填入置换表中并运行程序结果如下：

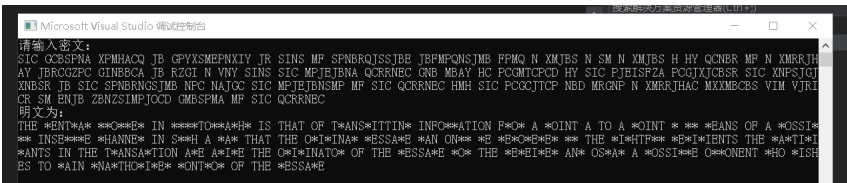


图 8: 一种可能的结果

可以看到，我们的猜测使得大部分字母都出来了，接下来可以根据单词进行字母的推断了。譬如，“INFORMATION”，“TRANSACTION”，“TRANSMITTING”(P推出R，Q推出M，G推出C，E推出G)等等，得到如下结果：

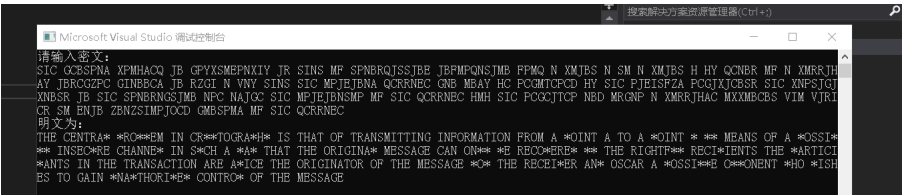


图 9: 已破译十三个字母

有趣的是，在思考的过程中，我发现了一个单词，它只有三个字母，其

对应的明文中，中间为‘O’(见图??)。而由密文可以看到，两边的单词是相同的，令人困惑的是，似乎没有什么单词具有这样的格式。在一段时间思考后，我们认为，这可能是一个人名，即“BOB”，加之密码学中经常出现的“ALICE”与“BOB”，我们推定，‘H’可能就是‘B’。

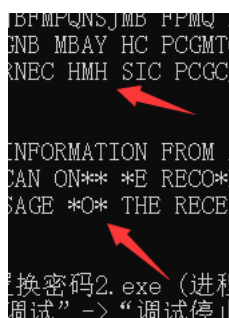
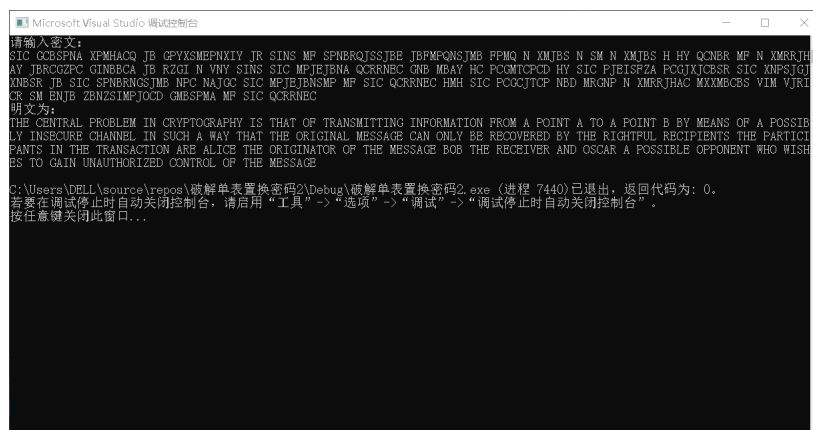


图 10: 不妨考虑为人名的情况

之后的推理便“一帆风顺”了，基本上根据单词的大致意思便能推出结果，下一页是推导过程：

$CENTRA* \Rightarrow CENTRAL$   
 $A \Rightarrow L$   
 $*ROBLEM \Rightarrow PROBLEM$   
 $X \Rightarrow P$   
 $*HO \Rightarrow WHO$   
 $*ISH \Rightarrow WISH$   
 $V \Rightarrow W$   
 $NBD \Rightarrow AN*$   
 $D \Rightarrow D$   
 $RECO*ERED \Rightarrow RECOVERED$   
 $T \Rightarrow V$   
 $B* \Rightarrow BY$   
 $ONL* \Rightarrow ONLY$   
 $Y \Rightarrow Y$   
 $S*CH \Rightarrow SUCH$   
 $RIGHTF*L \Rightarrow RIGHTFUL$   
 $Z \Rightarrow U$   
 $UNAUTHORI*ED \Rightarrow UNAUTHORIZED$   
 $O \Rightarrow Z$

至此，推导基本完全结束(事实上，还有四个字母未得出，但由于密文中仅出现了22个字母所导致的)。以下是填入置换表后程序运行的结果：



```
Microsoft Visual Studio 调试控制台
请输入密文:
SIC QGBSPMA XPMHACQ TB CPYXSMENXIV JR SINS MF SPNERQJSSSTBE JBFMPONGSMB FPMQ N XMIBS N SM N XMIBS H HY QCNBR MF N XMRJH
AY TBROCCPT GINEBCA TB RZGI N VNY SINS SIC MPTJBTENA QORRNEC GNB MBAY HQ PQMTCPQD HY SIC FJBISEZA FQJXJCESR SIC XNPSICJ
XNESR TB SIC SPNERGSSMB NPO NAJCC SIC MPTJBTNSMP MF SIC QORRNEC HMH SIC FQJCTJCP NBD MRCNP N XMRJHAC MXXMBES VIM VJRI
CR SM ENJB ZBNZSIMPJQD GMBSPMA MF SIC QORRNEC
明文为:
THE CENTRAL PROBLEM IN CRYPTOGRAPHY IS THAT OF TRANSMITTING INFORMATION FROM A POINT A TO A POINT B BY MEANS OF A POSSIB
LY INSECURE CHANNEL IN SUCH A WAY THAT THE ORIGINAL MESSAGE CAN ONLY BE RECOVERED BY THE RIGHTFUL RECIPIENTS THE PARTICI
PANTS IN THE TRANSACTION ARE ALICE THE ORIGINATOR OF THE MESSAGE BOB THE RECEIVER AND OSCAR A POSSIBLE OPPONENT WHO WISH
ES TO GAIN UNAUTHORIZED CONTROL OF THE MESSAGE
C:\Users\DELL\source\repos\破解单表置换密码2\Debug\破解单表置换密码2.exe (进程 7440) 已退出，返回代码为: 0。
若要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

图 11: 最终运行结果

最终明文：

THE CENTRAL PROBLEM IN CRYPTOGRAPHY IS THAT OF TRANSMITTING INFORMATION FROM A POINT A TO A POINT B BY MEANS OF A POSSIBLY INSECURE CHANNEL IN SUCH A WAY THAT THE ORIGINAL MESSAGE CAN ONLY BE RECOVERED BY THE RIGHTFUL RECIPIENTS THE PARTICIPANTS IN THE TRANSACTION ARE ALICE THE ORIGINATOR OF THE MESSAGE BOB THE RECEIVER AND OSCAR A POSSIBLE OPPONENT WHO WISHES TO GAIN UNAUTHORIZED CONTROL OF THE MESSAGE

最终置换表：

$$A \Rightarrow L$$

$$B \Rightarrow N$$

$$C \Rightarrow E$$

$$D \Rightarrow D$$

$$E \Rightarrow G$$

$$F \Rightarrow F$$

$$G \Rightarrow C$$

$$H \Rightarrow B$$

$$I \Rightarrow H$$

$$J \Rightarrow I$$

$$K \Rightarrow *$$

$$L \Rightarrow *$$

$$M \Rightarrow O$$

$$N \Rightarrow A$$

$$O \Rightarrow Z$$

$$P \Rightarrow R$$

$$Q \Rightarrow M$$

$$R \Rightarrow S$$

$$S \Rightarrow T$$

$$T \Rightarrow V$$

$$U \Rightarrow *$$

$$V \Rightarrow W$$

$$W \Rightarrow *$$

$$X \Rightarrow P$$

$$Y \Rightarrow Y$$

$$Z \Rightarrow U$$



## 附录

### A 移位密码算法

```
1      #include <iostream>
2      using namespace std;
3
4      char *global_text;
5
6      void encrypt(char *plaintext, int key)
7      {
8          for (int i = 0; i < strlen(plaintext); i++)
9          {
10             if (plaintext[i] >= 97 && plaintext[i] <= 122)
11             {
12                 int temp = plaintext[i];
13                 temp -= 32;
14                 plaintext[i] = (char)temp;
15             }
16             if (plaintext[i] >= 65 && plaintext[i] <= 90)
17             {
18                 int tmp = plaintext[i];
19                 tmp += key;
20                 tmp -= 65;
21                 tmp = tmp % 26;
22                 tmp += 65;
23                 plaintext[i] = (char)tmp;
24             }
25         }
26
27         cout << "The ciphertext is " << endl;
28
29         for (int i = 0; i < strlen(plaintext); i++)
30         {
31             cout << plaintext[i];
32         }
33
34         cout << endl;
35
36         global_text = plaintext;
37     }
38
39     void decrypt(char *ciphertext, int key)
40     {
41         for (int i = 0; i < strlen(ciphertext); i++)
42         {
43             if (ciphertext[i] >= 65 && ciphertext[i] <= 90)
44             {
45                 int tmp = ciphertext[i];
46                 tmp -= 65;
47                 tmp += (26 - key);
48                 tmp = tmp % 26;
49                 tmp += 65;
50                 ciphertext[i] = (char)tmp;
51             }
52         }
53         cout << "The plaintext is " << endl;
54
55         for (int i = 0; i < strlen(ciphertext); i++)
56         {
```

```

57         cout << ciphertext[i];
58     }
59     cout << endl;
60 }
61
62
63 int main()
64 {
65     char plaintext[1000] = { '\0' };
66     char ciphertext[1000] = { '\0' };
67
68     cout << "Please input the plaintext " << endl;
69     cin.getline(plaintext,1000);
70     int key;
71     cout << "please input the key " << endl;
72     cin >> key;
73
74     encrpty(plaintext , key);
75
76     memcpy(ciphertext , global_text , sizeof(ciphertext));
77
78     decrpty(ciphertext , key);
79
80     return 0;
81 }
82

```

## B 破解移位密码算法

```
1      #include <iostream>
2      using namespace std;
3
4
5      int decode(char *ciphertext)
6      {
7          int key;
8          for (int key = 1; key <= 25; key++)
9          {
10             char tmp_text[1000];
11             memcpy(tmp_text, ciphertext, sizeof(tmp_text));
12
13             for (int i = 0; i < strlen(tmp_text); i++)
14             {
15                 if (tmp_text[i] >= 97 && tmp_text[i] <= 122)
16                 {
17                     int temp = tmp_text[i];
18                     temp -= 32;
19                     tmp_text[i] = (char)temp;
20                 }
21                 if (tmp_text[i] >= 65 && tmp_text[i] <= 90)
22                 {
23                     int tmp = tmp_text[i];
24                     tmp -= 65;
25                     tmp += (26 - key);
26                     tmp = tmp % 26;
27                     tmp += 65;
28                     tmp_text[i] = (char)tmp;
29                 }
30             }
31
32             cout << "key=" << key << ",the plaintext is " << endl;
33
34             for (int i = 0; i < strlen(tmp_text); i++)
35             {
36                 cout << tmp_text[i];
37             }
38             cout << endl;
39             cout << endl;
40
41         }
42         return 0;
43     }
44
45
46
47
48
49     int main()
50     {
51         int asw;
52         char ciphertext[1000] = { '\0' };
53
54         cout << "Please input the ciphertext " << endl;
55         cin.getline(ciphertext, 1000);
56
57         cout << endl;
58
59         decode(ciphertext);
60     }
```

```
61 |         return 0;  
62 |     }  
63 |
```

## C 单表置换密码

```
1      #include<iostream>
2      #include<stdio.h>
3      #include<vector>
4      using namespace std;
5
6      char *global_text;
7      int len_text;
8
9      void encrypt(char *plaintext)
10     {
11         //char plaintext[1000] = { '\0' };
12         char ciphertext[1000] = { '\0' };
13         vector<char>key;
14         key.push_back('H');
15         key.push_back('K');
16         key.push_back('W');
17         key.push_back('T');
18         key.push_back('X');
19         key.push_back('Y');
20         key.push_back('S');
21         key.push_back('G');
22         key.push_back('B');
23         key.push_back('P');
24         key.push_back('Q');
25         key.push_back('E');
26         key.push_back('J');
27         key.push_back('A');
28         key.push_back('Z');
29         key.push_back('M');
30         key.push_back('L');
31         key.push_back('N');
32         key.push_back('O');
33         key.push_back('F');
34         key.push_back('C');
35         key.push_back('I');
36         key.push_back('D');
37         key.push_back('V');
38         key.push_back('U');
39         key.push_back('R');
40
41         for (unsigned int i = 0; i < strlen(plaintext); i++)
42         {
43             if (plaintext[i] >= 97 && plaintext[i] <= 122)
44             {
45                 int temp = plaintext[i];
46                 temp -= 32;
47                 plaintext[i] = (char)temp;
48             }
49             if (plaintext[i] >= 65 && plaintext[i] <= 90)
50             {
51                 int tmp = plaintext[i];
52                 tmp -= 65;
53                 ciphertext[i] = key[tmp];
54             }
55             else
56             {
57                 ciphertext[i] = plaintext[i];
58             }
59         }
60     }
```

```

61         cout << "The ciphertext is " << endl;
62
63         for (unsigned int i = 0; i < strlen(plaintext); i++)
64         {
65             cout << ciphertext[i];
66         }
67
68         cout << endl;
69
70         global_text = ciphertext;
71
72     }
73
74     char find(char cipher)
75     {
76         vector<char>key;
77         key.push_back('H');
78         key.push_back('K');
79         key.push_back('W');
80         key.push_back('T');
81         key.push_back('X');
82         key.push_back('Y');
83         key.push_back('S');
84         key.push_back('G');
85         key.push_back('B');
86         key.push_back('P');
87         key.push_back('Q');
88         key.push_back('E');
89         key.push_back('J');
90         key.push_back('A');
91         key.push_back('Z');
92         key.push_back('M');
93         key.push_back('L');
94         key.push_back('N');
95         key.push_back('O');
96         key.push_back('F');
97         key.push_back('C');
98         key.push_back('I');
99         key.push_back('D');
100        key.push_back('V');
101        key.push_back('U');
102        key.push_back('R');
103        for (int i = 0; i <= 25; i++)
104        {
105            if (cipher == key[i])
106            {
107                int tmp = i;
108                tmp += 65;
109                cipher = (char)tmp;
110                break;
111            }
112        }
113        return cipher;
114    }
115
116    void decrypt(char *ciphertext)
117    {
118        cout << "The plaintext is " << endl;
119
120        for (unsigned int i = 0; i < len_text; i++)
121        {
122            ciphertext[i] = find(ciphertext[i]);
123            cout << ciphertext[i];

```

```

124         }
125
126         cout << endl;
127     }
128
129     int main()
130     {
131         char plaintext[1000] = { '\0' };
132         char ciphertext[1000] = { '\0' };
133
134         cout << "Please input the plaintext ";
135
136         cin.getline(plaintext, 1000);
137
138         len_text = strlen(plaintext);
139
140         encrypt(plaintext);
141
142         memcpy(ciphertext, global_text, sizeof(ciphertext));
143
144         decrypt(ciphertext);
145
146         return 0;
147     }
148
149
150

```

## D 字母频率统计程序

```
1      #include <iostream>
2      #include <map>
3      using namespace std;
4
5      int main()
6      {
7          double pra = 0;
8          int sum = 0;
9          double my_sort[30];
10         char text[1000] = { '\0' };
11         map<int, double>freq;
12
13         cin.getline(text, 1000);
14
15         for (int i = 0; i < 26; i++)
16         {
17             freq[i] = 0.0;
18         }
19
20         for (int i = 0; i < strlen(text); i++)
21         {
22             if (text[i] >= 65 && text[i] <= 90)
23             {
24                 sum++;
25             }
26         }
27
28         for (int i = 0; i < strlen(text); i++)
29         {
30             if (text[i] >= 65 && text[i] <= 90)
31             {
32                 int temp = text[i];
33                 temp -= 65;
34                 freq[temp] = freq[temp] + 1;
35             }
36         }
37
38         for (int i = 0; i < 26; i++)
39         {
40             int temp = 65 + i;
41             char tmp = (char)temp;
42             freq[i] = freq[i]/sum;
43             pra += freq[i];
44             my_sort[i] = freq[i] * 100;
45             cout << tmp << "=" << freq[i] * 100 << endl;
46         }
47     }
48
49     for (int i = 0; i < 26; i++)
50     for (int j = 0; j < i; j++)
51     {
52         if (my_sort[i] > my_sort[j])
53         {
54             double my_temp = my_sort[i];
55             my_sort[i] = my_sort[j];
56             my_sort[j] = my_temp;
57         }
58     }
59
60     for (int i = 0; i < 26; i++)
```



```
61         {
62             cout << my_sort[i] << endl;
63         }
64
65         cout << "pra=" << pra << endl;
66
67         return 0;
68     }
69
70
71
```

## E 密文破解所用程序

```
1      #include<iostream>
2      #include<stdio.h>
3      #include<vector>
4      using namespace std;
5
6      void decrypt(char *plaintext)
7      {
8          //char plaintext[1000] = { '\0' };
9          char ciphertext[1000] = { '\0' };
10         vector<char>key;
11         key.push_back('L');//A    L
12         key.push_back('N');//B    N
13         key.push_back('E');//C    E *
14         key.push_back('D');//D    D
15         key.push_back('G');//E    G
16         key.push_back('F');//F    F
17         key.push_back('C');//G    C
18         key.push_back('B');//H    B
19         key.push_back('H');//I    H *
20         key.push_back('I');//J    I
21         key.push_back('*');//K
22         key.push_back('*');//L
23         key.push_back('O');//M    O
24         key.push_back('A');//N    A *
25         key.push_back('Z');//O    Z
26         key.push_back('R');//P    R
27         key.push_back('M');//Q    M
28         key.push_back('S');//R    S
29         key.push_back('T');//S    T *
30         key.push_back('V');//T    V
31         key.push_back('*');//U
32         key.push_back('W');//V    W
33         key.push_back('*');//W
34         key.push_back('P');//X    P
35         key.push_back('Y');//Y    Y
36         key.push_back('U');//Z    U
37
38         for (unsigned int i = 0; i < strlen(plaintext); i++)
39         {
40             if (plaintext[i] >= 97 && plaintext[i] <= 122)
41             {
42                 int temp = plaintext[i];
43                 temp -= 32;
44                 plaintext[i] = (char)temp;
45             }
46             if (plaintext[i] >= 65 && plaintext[i] <= 90)
47             {
48                 int tmp = plaintext[i];
49                 tmp -= 65;
50                 ciphertext[i] = key[tmp];
51             }
52             else
53             {
54                 ciphertext[i] = plaintext[i];
55             }
56         }
57
58         cout << "The plaintext is " << endl;
59
60         for (unsigned int i = 0; i < strlen(plaintext); i++)
```

```

61         {
62             cout << ciphertext[i];
63         }
64
65         cout << endl;
66
67     }
68
69
70     int main()
71     {
72         char plaintext[1000] = { '\0' };
73         char ciphertext[1000] = { '\0' };
74
75         cout << "Please input the ciphertext " << endl;
76
77         cin.getline(plaintext, 1000);
78
79         decrypt(plaintext);
80
81         return 0;
82     }
83

```

## F 移位密码算法流程图

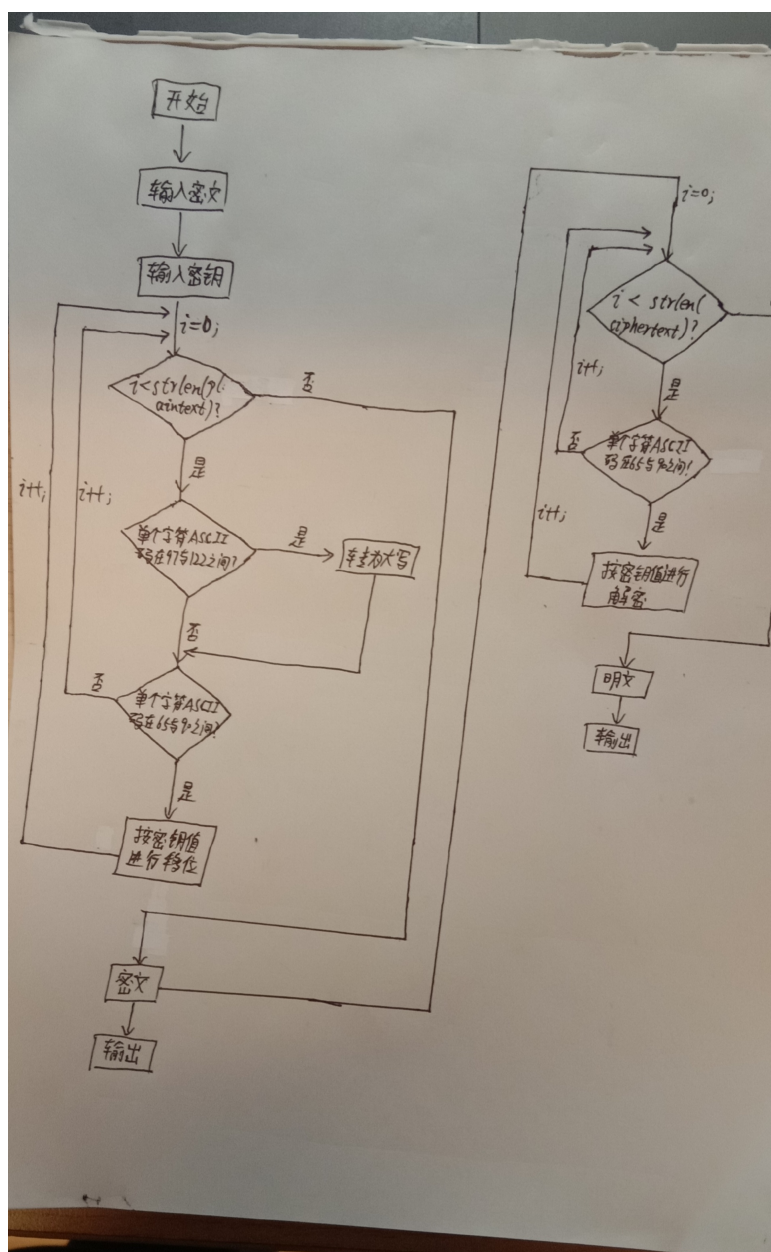


图 12: 移位密码流程图(手绘)

## G 单表置换密码算法流程图

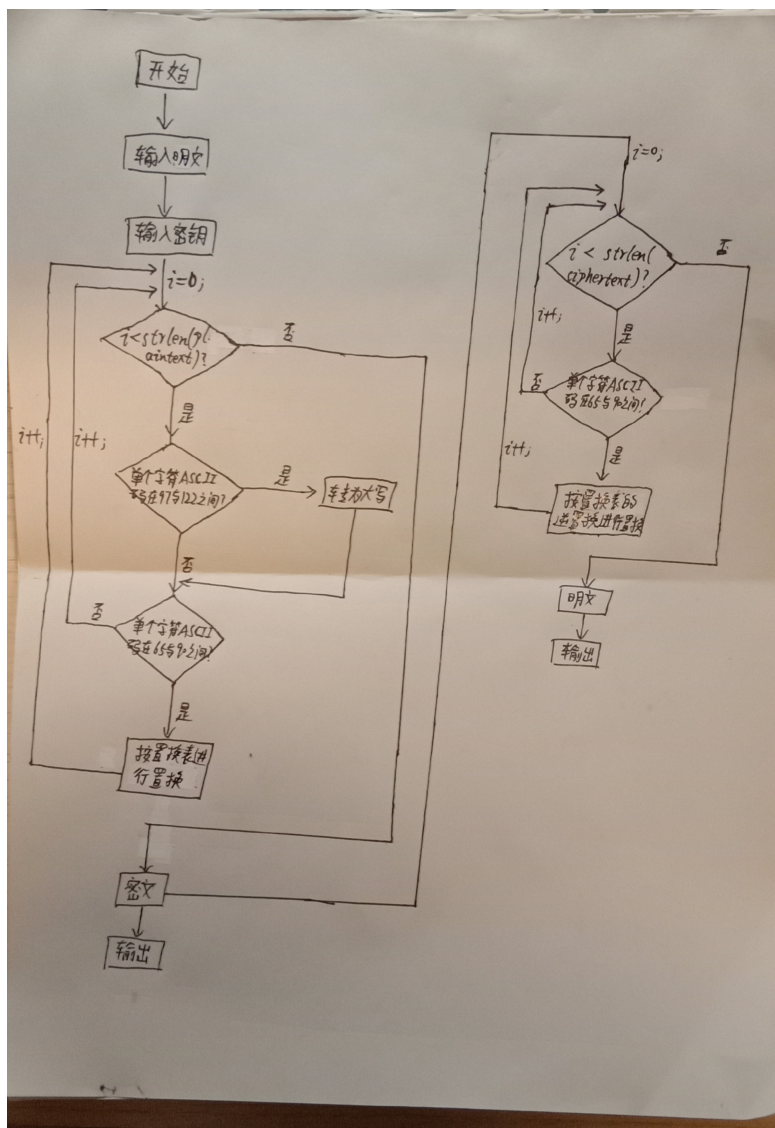


图 13: 单表置换密码流程图(手绘)