
Contents

阅读须知	3
任务一	6
个位求和【增强版】	6
思路	6
代码	7
找顺数（加强版）	7
思路	7
代码	7
求对称数	8
思路	8
代码	8
通讯录排序	9
思路	9
代码	9
主元素	10
思路	10
代码	10
任务 2-隐私信息管理系统	11
需求分析	11
设计目标	11
总体设计	11
编译环境	11
文件模块介绍	12
详细设计	12
src/main.cpp	12
data.h/data.cpp 简要介绍	13
ui.h	17
os.hpp	17
mods 文件夹下功能模块	17
遇到的问题和解决办法	19
结束语	20
任务三-电子印章制作程序	20
需求分析	20

设计目标	20
总体设计	20
编译环境	20
文件模块介绍	21
详细设计	21
bmp.hpp	21
image.hpp	22
fonts.hpp	23
conv.hpp	24
main.cpp	24
xmake.lua	26
遇到的问题与解决方法	26
结束语	27

阅读须知

本次课设报告任务一的代码前均引入了 `ATL/base/include/cardinal.hpp` 头文件，为避免重复在这里一次性给出。

任务二和任务三将使用 `xmake` 构建工具，使用 `C++20` 标准编译。

本人使用 `C++11` 高级模板技巧原创 `ATL` 通用算法模板库 `github` 开源地址: <https://github.com/OrbitZore/ATL>

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  struct FAST_IO{
4      FAST_IO(){
5          ios_base::sync_with_stdio(false);
6          cin.tie(NULL);
7      }
8  }__fast_io;
9  #if __cplusplus < 201402L
10 template<class T, class U=T>
11 T exchange(T& obj, U&& new_value){
12     T old_value=move(obj);
13     obj=forward<U>(new_value);
14     return old_value;
15 }
16 #endif
17 #define cons(a,...) a=typename decay<decltype(a)>::type(__VA_ARGS__)
18 using INT=int;
19 #define x first
20 #define y second
21 // #define int long long
22 #define pb push_back
23 #define eb emplace_back
24 #define all(a) (a).begin(),(a).end()
25 auto &_ =std::ignore;
26 using ll=long long;
27 template<class T>
28 using vec=vector<T>;
29 template<bool B,class T=void>
30 using enableif_t=typename enable_if<B,T>::type;
31
32 #define DEF_COULD(name,exp) \
33 template<class U> \
34 struct name{\
35     template<class T>\
36     constexpr static auto is(int i)->decltype(exp,true){return true;}\
37     template<class T>\
38     constexpr static bool is(...){return false;}\
39     static const bool value=is<U>(1);\
40 };
41 #define DEF_CAN(name,exp) DEF_COULD(can##name,exp)
42 #define ENABLE(T,name) enableif_t<can##name<T>::value>(1)
43 #define ENABLEN(T,name) enableif_t<!can##name<T>::value>(1)
44 #define FOR_TUPLE enableif_t<i!=tuple_size<T>::value>(1)
45 #define END_TUPLE enableif_t<i==tuple_size<T>::value>(1)
46 #define FOR_TUPLET(T) enableif_t<i!=tuple_size<T>::value>(1)
```

```

47 #define END_TUPLE(T) enable_if_t<i==tuple_size<T>::value>(1)
48
49 #define DEF_INF(name,exp)\
50 constexpr struct{\
51     template<class T>\
52     constexpr operator T()const {return numeric_limits<T>::exp();}\
53 } name;
54
55 DEF_CAN(Out,(cout<<*(T*)(0))) DEF_CAN(For,begin(*(T*)(0)))
56 DEF_INF(INF,max) DEF_INF(MINF,min)
57
58 template<size_t i,class T>
59 auto operator>>(istream& is,T &r)->decltype(END_TUPLE,is){
60     return is;
61 }
62 template<size_t i=0,class T>
63 auto operator>>(istream& is,T &r)->decltype(FOR_TUPLE,is){
64     is>>get<i>(r);
65     return operator>> <i+1>(is,r);
66 }
67
68 template<size_t i,class ...Args>
69 auto operator>>(istream& is,const tuple<Args&...> &r)->decltype(END_TUPLE(T,
    tuple<Args&...>),is){
70     return is;
71 }
72 template<size_t i=0,class ...Args>
73 auto operator>>(istream& is,const tuple<Args&...> &r)->decltype(FOR_TUPLE(T,
    tuple<Args&...>),is){
74     is>>get<i>(r);
75     return operator>> <i+1>(is,r);
76 }
77
78 template<class T>
79 auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLE(T,Out),c
    +1);
80 template<size_t i,class T>
81 auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLEN(T,For),
    END_TUPLE,c+1);
82 template<size_t i=0,class T>
83 auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLEN(T,For),
    FOR_TUPLE,c+1);
84 template<class T>
85 auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLEN(T,Out),
    ENABLE(T,For),c+1);
86
87
88 template<class T>
89 auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLE(T,Out),c
    +1){
90     os << cv;
91     while (*c!='}') c++;
92     return c+1;
93 }
94 template<size_t i,class T>
95 auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLEN(T,For),

```

```

115         END_TUPLE,c+1){
116         return c;
117     }
118     template<size_t i,class T>
119     auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLEN(T,For),
120         FOR_TUPLE,c+1){
121         while (*c!='{') os << *c++;
122         c=__format(os,c,get<i>(cv));
123         return __format<i+1>(os,c,cv);
124     }
125     template<class T>
126     auto __format(ostream &os,const char *c,const T& cv)->decltype(ENABLEN(T,Out),
127         ENABLE(T,For),c+1){
128         const char *ct=c+1;
129         if (cv.size()==0){
130             while (*ct!='}') ct++;
131             ct++;
132             while (*ct!='}') ct++;
133         }else{
134             for (auto &i:cv){
135                 const char *cc=c+1;
136                 while (*cc!='{') os << *cc++;
137                 cc=__format(os,cc,i);
138                 while (*cc!='}') os << *cc++;
139                 ct=cc;
140             }
141         }
142         return ct+1;
143     }
144     void _format(ostream &os,const char *c){
145         while (*c!='{'&&*c!='\0') os<< *c++;
146     }
147     template<class T,class ...Args>
148     void _format(ostream &os,const char *c,const T &a,const Args& ...rest){
149         while (*c!='{'&&*c!='\0') os<< *c++;
150         if (*c=='{') c=__format(os,c,a);
151         _format(os,c,rest...);
152     }
153     template<class ...Args>
154     string format(const char *c,const Args& ...rest){
155         ostringstream os;
156         _format(os,c,rest...);
157         return os.str();
158     }
159     template<class ...Args>
160     ostream& print(const char *c,const Args& ...rest){return _format(cout,c,rest
161         ...),cout;}
162     template<class ...Args>
163     ostream& println(const char *c,const Args& ...rest){return print(c,rest...)<<
164         endl;}
165
166     #ifndef LOCAL
167     #define debug(...) cerr<<format(__VA_ARGS__)
168     #else
169     #define debug(...) cerr
170     #endif

```

```

147 template<class T,class ...Args>
148 struct Rtar{
149     T& a;tuple<Args...> n;
150     Rtar(T& a,tuple<Args...> n):a(a),n(n){}
151 };
152 template<class T,class ...Args>
153 Rtar<T,Args&...> rtar(T &a,Args&... rest){
154     return Rtar<T,Args&...>(a,tie(rest...));
155 }
156 template<size_t i,class U,class ...Args,class T=tuple<Args&...>>
157 auto operator>>(istream& is,Rtar<U,Args&...> r)->decltype(END_TUPLE,is){
158     return is>>r.a;
159 }
160 template<size_t i=0,class U,class ...Args,class T=tuple<Args&...>>
161 auto operator>>(istream& is,Rtar<U,Args&...> r)->decltype(FOR_TUPLE,is){
162     r.a=typename decay<U>::type(get<i>(r.n));
163     for (auto &w:r.a)
164         operator>> <i+1>(is,Rtar<decltype(w),Args&...>(w,r.n));
165     return is;
166 }
167 template<class T1,class T2>
168 bool cmin(T1 &a,const T2 b){return a>b?a=b,1:0;}
169 template<class T1,class T2>
170 bool cmax(T1 &a,const T2 b){return a<b?a=b,1:0;}
171 template<class T1,class T2,class ...T3>
172 bool cmin(T1 &a,const T2 b,const T3 ...rest){return cmin(a,b)|cmin(a,rest...);}
173 template<class T1,class T2,class ...T3>
174 bool cmax(T1 &a,const T2 b,const T3 ...rest){return cmax(a,b)|cmax(a,rest...);}
175 bool MULTIDATA=true;

```

任务一

个位求和【增强版】

将一个整数区间内所有整数的个位相加并输出。

$[m, n](m \leq n)$

思路

- 不妨设 $m \geq 0$ 设 $f(a)$ 为 $[1, a]$ 中所有整数的个位数和，答案即为 $f(n) - f(m - 1)$ 。观察到 $f(a+10) = f(a) + \sum_{i=0}^9 i = f(a) + 45$ ，可知 $f(a) = f(a\%10) + 45 \cdot \lfloor a/10 \rfloor$ 预处理出 $f(a), a \in [0, 9]$ 即可。
- 考虑 $m < 0$
 - $n < 0$ 等价于 $n = -n, m = -m$ 的情况。
 - $n > 0$ 等于 $f(n) + f(-m)$ 。

代码

```
1 constexpr int dsum(int b,int c,int n){
2     return (2*b+c*(n-1))*n/2;
3 }
4
5 constexpr int ssum(int b,int e,int n){
6     return (b+e)*n/2;
7 }
8
9 ll f(int i){
10     i=abs(i);
11     return i/10*ssum(0,9,10)+dsum(0,1,i%10+1);
12 }
13
14 int main(){
15     ll n,m;
16     cin>>n>>m;
17     if (n>0&& m>0 || n<0&& m<0){
18         tie(n,m)=minmax({abs(n),abs(m)});
19         n--;
20         cout<<(f(m)-f(n))<<endl;
21     }else cout << f(n)+f(m);
22 }
```

找顺数（加强版）

现在给出一个大于1的正整数 n ，请你求出在1到 n 的正整数中，至少有一个数位含有6的正整数个数。

思路

设 n 有 nb 位。

定义 $f(a)$ 为 a 位数的个数。

$g(a)$ 为 nb 位数的后 $a-1$ 位中至少有一个数位含有6，而且第 nb 位到 a 位和 n 相等的整数个数。

递归处理即可。

代码

```
1 int c[100];
2 int cn[100];
3 int p10[10];
4 int F[10];
5 int f(int i){
6     if (F[i]!=-1) return F[i];
```

```

7     if (i==1) return F[i]=1;
8     return F[i]=p10[i-1]+9*f(i-1);
9 }
10 int g(int n){
11     int cc=c[n-1];
12     if (n==1)
13         return cc>=6?1:0;
14     if (cc==6)
15         return cc*f(n-1)+cn[n-2]+1;
16     if (cc>6)
17         return (cc-1)*f(n-1)+p10[n-1]+g(n-1);
18     return cc?cc*f(n-1)+g(n-1):g(n-1);
19 }
20 int main(){
21     memset(F,-1,sizeof(F));
22     p10[0]=1;for (int i=1;i<10;i++) p10[i]=10*p10[i-1];
23     int n=0;
24     while (isdigit(c[n]=cin.get())) c[n++]!='0';
25     c[n]=0;
26     for (int i=0;i<n/2;i++) swap(c[i],c[n-1-i]);
27     cn[0]=c[0];
28     for (int i=1;i<n;i++)
29         cn[i]=c[i]+cn[i-1]*10;
30     cout << g(n) << endl;
31 }

```

求对称数

如果给定一个对称数 n ，请你求出大于 n 的最小对称数 (即这个数从左向右读和从右向左读是完全一样的)。

$$n \leq 10^9$$

思路

考察全体小于 10^9 对称数集合 A 的大小，发现对于位数 m 为偶数的对称数可以建立其与全体位数为 $\frac{m}{2}$ 的自然数的双射。对于位数 m 为奇数的对称数可以建立其与全体位数为 $\lceil \frac{m}{2} \rceil$ 的自然数的双射。

所以 A 的大小在 10^5 左右，尝试生成 A ，二分解决。

代码

```

1  vec<int> s;
2  int p10[10];
3  void f(int n,int i=0,int w=0){
4      if (i>(n-1)/2){
5          for (;i!=n;i++){
6              w+=p10[i]*(w/p10[n-1-i]%10);

```



```

7         }
8         s.push_back(w);
9         return ;
10    }
11    for (int j=i==0?1:0;j<10;j++)
12        f(n,i+1,w+p10[i]*j);
13 }
14 int main(){
15     p10[0]=1;for (int i=1;i<10;i++) p10[i]=10*p10[i-1];
16     for (int i=1;i<10;i++){
17         f(i);
18     }
19     f(10,1,1);
20     sort(all(s));
21     int T;cin>>T;
22     while (T--){
23         int n;cin>>n;
24         cout<<*upper_bound(all(s),n)<<endl;
25     }
26 }

```

通讯录排序

输入 n 个朋友的信息，包括姓名、生日、电话号码，本题要求编写程序，按照年龄从大到小的顺序依次输出通讯录。题目保证所有人的生日均不相同。

思路

使用 ATL 处理读入，传比较年龄 lambda 给 `std::sort`。注意年龄使用字符串存，数字存需要处理前导零。

代码

```

1  int main(){
2      int n;
3      using T=tuple<string,array<char,8>,string>;
4      vec<T> v;
5      cin>>n>>rtar(v,n);
6      sort(all(v),[](const T& x,const T& y){
7          return get<1>(x)<get<1>(y);
8      });
9      print("{} {} {} \n",v);
10 }

```

主元素

已知一个整数序列 $A = (a_0, a_1, \dots, a_{n-1})$ 。若存在 $a_{p_1} = a_{p_2} = \dots = a_{p_m} = x$ 且 $m > \frac{n}{2}$ ($0 \leq p_k < n, 1 \leq k \leq m$)，则称 x 为 A 的主元素。

例如 $A=(0, 5, 5, 3, 5, 7, 5, 5)$ ，则 5 为主元素；又如 $A=(0, 5, 5, 3, 5, 1, 5, 7)$ ，则 A 中没有主元素。

$n \leq 100000$

求主元素，如果没有主元素则输出 -1。

思路

- 假设有主元素。

考虑随机选中序列 A 中的一个元素，有大于一半的概率选中主元素。那么任取两元素 a, b ，只要一个不是主元素，删掉后序列的主元素不变。

- 没有主元素

使用有主元素的方法求出，然后再遍历一遍 A 判断出现次数即可。

代码

```
1  int main(){
2      int n;
3      vec<int> v;
4      while (cin>>n>>rtar(v,n)){
5          int w=-1,wc=0;
6          for (auto i:v)
7              if (i!=w){
8                  if (wc) wc--;
9                  else {w=i;wc=1;}
10             }else wc++;
11         int c=0;
12         for (auto i:v) if (w==i) c++;
13         if (c>n/2) cout << w << endl;
14         else cout << -1 << endl;
15         v.clear();
16     }
17 }
```

任务 2-隐私信息管理系统

需求分析

用户需要一个软件保存网站上保存的用户名和密码。需要保存的记录如下表所示。

编号	帐号位置	帐号描述	帐号名	密码
1	https://next.xuetangx.com	学堂在线	xiangdesheng	123456
2	http://acm.hdu.edu.cn	杭电 OJ	acm002	654321
3	https://www.icourse163.org	中国大学 MOOC	30047495@qq.com	123456

设计目标

系统具体提供以下功能：

- 系统以菜单方式工作。开始运行程序时要进行密码验证。
- 信息的录入功能、浏览功能。
- 信息的查询功能。按帐号名查询，如输入“acm”可查出上表第 2 条记录。
- 信息的删除、修改功能。
- 信息存入文件。信息中的密码必须要加密后才能存入文件。

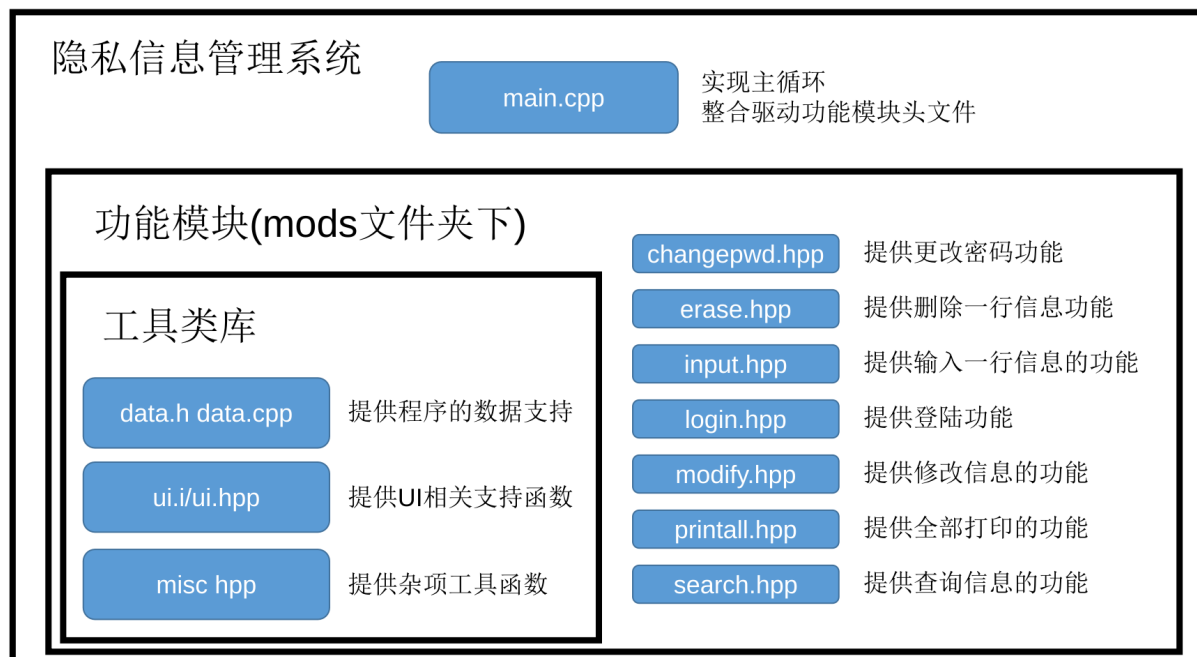
除此以外，因选定的编译环境支持多操作系统，所以本项目亦需要支持 windows 和 linux 操作系统。

总体设计

编译环境

本次实验将使用头文件和单头文件技术。使用 xmake 编译工具，C++20 标准编译项目。

文件模块介绍



详细设计

src/main.cpp

```
1  #include "cardinal.hpp"
2  #include "ui.h"
3  #include "data.h"
4  #include "mods/allmods.hpp"
5
6  const vec<string> main_menu={
7      "输入信息",
8      "查询信息",
9      "查看全部信息",
10     "修改信息",
11     "删除信息",
12     "更改密码",
13     "退出"
14 };
15 void(*menu_func[])(void)={
16     input,
17     search,
18     printall,
19     modify,
20     erase,
21     changepwd,
22     [](){exit(0);}
```

```

23 };
24 int main(int argc, char** argv){
25     inidb1();//data.h提供, 初始化密码
26     login();//mod/login.hpp提供
27     inidb2();//data.h提供,
28     while (true){//主循环
29         int w=ui::choose(main_menu,"主菜单");
30         menu_func[w]();
31         syncdb();
32         ui::getline("按回车键返回主菜单",true);
33     }
34     return 0;
35 }

```

data.h/data.cpp 简要介绍

- data.h

```

1  string encrypt(const string& a,const string& key="");//加密
2  string decrypt(const string& a,const string& key="");//解密
3  string shash(const string& a);//哈希
4
5  string encode(const string& a);//编码 (去除换行符)
6  string decode(const string& a);//解码
7
8  inline const map<string,string> default_kv={//默认密码
9      {"password",shash("passwd")},
10 };
11 struct user{//一行信息
12     string pos,description,name,pwd;
13     string to_string() const;//序列化
14     static user from_string(string str);//反序列化
15 };
16 using dbT=map<string,string>;//键值数据库基于std::map
17 extern dbT data_base;//键值数据库
18 extern vec<user> user_list;//用户信息列表
19 void syncdb();//写入数据库到文件
20
21 void inidb1();//初始化密码
22 void inidb2();//登陆成功后获取数据库

```

- data.cpp

加密/解密函数实现使用了模意义上的乘法逆元，加密步骤循环对原串每一位在模数为 65521 的同余系上乘上 a ，以两个字节输出。 a 每次和 `key` 迭代加盐哈希。解密则反之。能实现较强的加密，但是会占用两倍的存储空间。

```

1  string encrypt(const string& a,const string& key){
2      using mchar=_mint<uint16_t,uint32_t,65521>;
3      auto num=hash<string>()(key);
4      string w;
5      for (auto i:a){

```

```

6         mchar c=(uint16_t)(unsigned char)i;
7         c*=(mchar)num;
8         w+=c.v&(0x00ff);
9         w+=(c.v&(0xff00))>>8;
10        num=hash<string>()(to_string(num)+"salt"+key);
11    }
12    return w;
13 }
14
15 string decrypt(const string& a,const string& key){
16     using mchar=_mint<uint16_t,uint32_t,65521>;
17     auto num=hash<string>()(key);
18     string w;
19     for (size_t i=0;i<a.size();i+=2){
20         mchar c=(uint16_t)(unsigned char)a[i]|
21             ((uint16_t)(unsigned char)a[i+1]<<8);
22         c/=(mchar)num;
23         w+=(char)c.v;
24         num=hash<string>()(to_string(num)+"salt"+key);
25     }
26     return w;
27 }

```

为了方便哈希直接使用了 `std::hash`。

```

1 string shash(const string& a){return to_string(hash<string>()(a));}

```

`encode`和`decode`函数用于转义和反转义。转义的目的是为了去掉回车符。而`decode`使用转义后没有单个' \ '字符的特性，将输入字符串根据单个' \n ' 字符分割输入字符串。可以用于用户信息序列化字符串的分割和多用户信息的分割。

```

1 string encode(const string& a){//del \n
2     string r;
3     for (auto &i:a){
4         if (i=='\\'){
5             r+="\\";
6         }else if (i=='\n'){
7             r+="\n";
8         }else r+=i;
9     }
10    return r;
11 }
12
13 string decode(const string& a){//gen \n
14     string r;
15     for (size_t i=0;i<a.size();i++){
16         auto ai=a[i];
17         if (ai=='\\'){
18             if (i+1<a.size()){
19                 if (a[i+1]=='n'){
20                     i++;
21                     r+='\n';
22                 }else if (a[i+1]=='\\'){
23                     i++;
24                     r+='\\';

```

```

25         }
26     }
27     }else r+=ai;
28 }
29 return r;
30 }
31
32 vec<string> decodep(const string& a){//gen \n
33     vec<string> r;string w;
34     for (size_t i=0;i<a.size();i++){
35         auto ai=a[i];
36         if (ai=='\\'){
37             if (i+1<a.size()){
38                 if (a[i+1]=='n'){
39                     i++;
40                     w+='\\n';
41                 }else if (a[i+1]=='\\'){
42                     i++;
43                     w+='\\';
44                 }else{
45                     if (w.size()){
46                         r.push_back(move(w));
47                         w=string();
48                     }
49                 }
50             }
51         }else w+=ai;
52     }
53     if (w.size()) r.push_back(move(w));
54     return r;
55 }

```

单个用户数据的序列化和反序列化，和全部用户数据的序列化和反序列化。

```

1  #define config_filename (config_dir+"pwdkeeper.cfg")
2
3  string user::to_string()const{
4      return
5          encode(pos)+"\\"+
6          encode(description)+"\\"+
7          encode(name)+"\\"+
8          encode(pwd);
9  }
10
11  user user::from_string(string str){
12      user u;
13      auto v=decodep(str);
14      u.pos=move(v.at(0));
15      u.description=move(v.at(1));
16      u.name=move(v.at(2));
17      u.pwd=move(v.at(3));
18      return u;
19  }
20
21  string encode_users(const vec<user>& v){
22      string str;

```

```

23     for (size_t i=0;i<v.size();i++)
24         str+=encode(v[i].to_string())+"\\\";
25     return str;
26 }
27
28 vec<user> decode_users(const string& str){
29     vec<user> v;
30     auto w=decodep(str);
31     for (auto& i:w)
32         v.push_back(user::from_string(i));
33     return v;
34 }

```

数据库存入文件时将用户信息列表存入数据库，然后将数据库序列化后加密输出。

```

1 void syncdb(){
2     ofstream os(config_filename,ios::binary);
3     string out;
4     data_base["users"]=encode_users(user_list);
5     for (auto &[k,v]:data_base){
6         out+=encode(k)+"="+encode(v)+"\n";
7     }
8     os<<data_base["password"]<<endl<<
9     encrypt(out,decrypt_key);
10 }

```

数据库的读入分两个步骤，第一步调用 `inidb1` 获取到密码的哈希值。之后 `login` 模块会根据密码的哈希值计算得到数据库加密密钥 `decrypt_key`。第二步调用 `inidb2` 根据 `decrypt_key` 解密数据库。

```

1 void inidb1(){
2     data_base=default_kv;
3     ifstream is(config_filename,ios::binary);
4     string a;
5     getline(is,a);
6     if (a.size()){
7         data_base["password"]=a;
8     }else return ;
9 }
10 void inidb2(){
11     string input;
12     {
13         ifstream is(config_filename,ios::binary);
14         while (is&&is.get()!='\n');
15         while (is) input+=is.get();
16     }
17     auto w=split(decrypt(input,decrypt_key),"\n");
18     for (auto& str:w){
19         int w=str.find('=');
20         if (str.size()&&(0<=w&&w<str.size()))
21             data_base[decode(str.substr(0,w))]=decode(str.substr(w+1));
22     }
23     user_list=decode_users(data_base["users"]);
24 }

```

ui.h

所有 ui 被包含在 ui 命名空间下，基于 UI 和数据操作可以分离的思想（即前后端分离）。

```
1  #pragma once
2
3  #include "cardinal.hpp"
4  namespace ui{
5
6  string getline(const string& msg,bool allow_empty=false);//根据提示请求用户输入
    一行字符串
7
8  istream& getline(string &str,const string& msg,bool allow_empty=false);//上一个
    函数的引用版本
9
10 int choose(const vec<string>& v,const string& name="");//在菜单中选择一项
11
12 size_t getint(size_t l,size_t r,const string& msg="");//输入[l,r)中的一个整数
13 }
```

os.hpp

根据操作系统的不同使用不同路径保存数据库。

```
1  #ifdef _WIN32
2  inline const string config_dir=string(getenv("appdata"))+"\\\\";
3  #else
4  inline const string config_dir=string(getenv("HOME"))+"/.config/";
5  #endif
```

mods 文件夹下功能模块

changepwd.hpp 更改密码，为了安全写入到数据库的是密码的哈希值。

```
1  inline void changepwd(){
2      string pwdnew=ui::getline("请输入新密码:"s);
3      data_base["password"]=shash(pwdnew);
4      update_decrypt_key(pwdnew);
5      println("更改密码成功");
6  }
```

earse.hpp 删除一行信息。

```
1  inline void erase(){
2      size_t wi=ui::getint(0,user_list.size(),"请输入将删除的编号");
3      auto& u=user_list[wi];
4      println("选中) {} {} {} {} {} {} {} ",wi,u.pos,u.description,u.name,u.pwd);
5      string str=ui::getline("输入yes (小写) 删除");
6      if (str=="yes"){
```

```

7         user_list.erase(user_list.begin()+wi);
8         println("删除完毕");
9     }else{
10        println("用户取消删除");
11    }
12    syncdb();
13 }

```

input.hpp 添加一个新用户。

```

1 inline void input(){
2     user_list.push_back(user{
3         ui::getline("请输入新用户帐号位置"),
4         ui::getline("请输入新用户帐号描述"),
5         ui::getline("请输入新用户帐号名"),
6         ui::getline("请输入新用户密码")
7     });
8     syncdb();
9 }

```

login.hpp 登陆模块。

```

1 inline string decrypt_key;
2 inline void update_decrypt_key(const string& rawpwd){
3     decrypt_key=shash(rawpwd+"salt"+shash(rawpwd)); //加盐
4 }
5 inline void login(){
6     while (1){
7         string password;
8         ui::getline(password,"请输入密码:s");
9         if (data_base["password"]==shash(password)){
10             println("登陆成功");
11             update_decrypt_key(password);
12             return false;
13         }
14         println("用户名或密码错误，登陆失败");
15         return true;
16     }();
17 }

```

modify.hpp 修改一个用户信息的模块。

```

1 inline void modify(){
2     size_t wi=ui::getint(0,user_list.size(),"请输入将修改的编号");
3     auto& u=user_list[wi];
4     println("选中) {} {} {} {} {} {} {} ",wi,u.pos,u.description,u.name,u.pwd);
5     size_t ci=ui::choose({
6         "帐号位置","帐号描述","帐号名","密码"
7     }, "请选中将要修改的值");
8     if (ci==0){
9         u.pos=ui::getline("请输入新的帐号位置");

```

结束语

学习到了现代 C++ 工程构建技术，和玩具级加密解密技术。

任务三-电子印章制作程序

需求分析

用户需要制作一个软件输入四字生成以下形状的电子印章。



设计目标

程序具体要实现以下功能

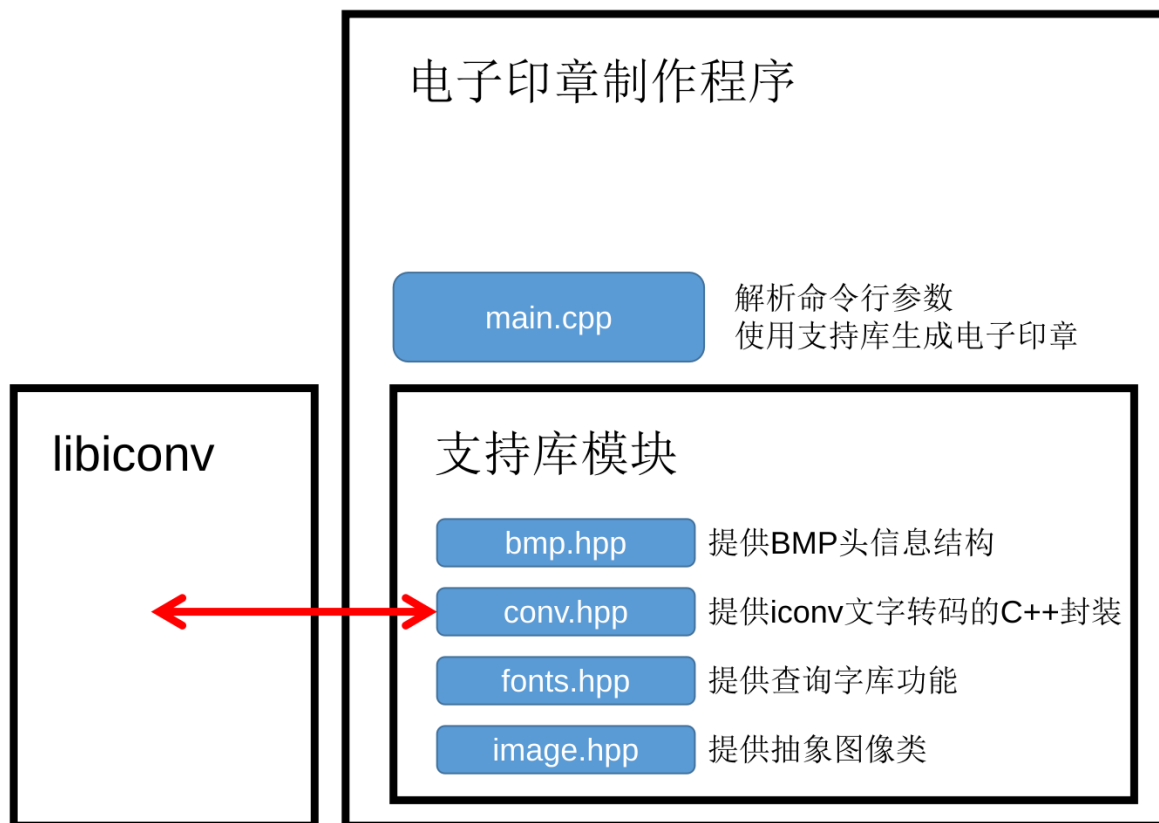
- 读取字库
- bmp 文件格式的写入

总体设计

编译环境

本次实验将使用头文件和单头文件技术。使用 `xmake` 编译工具，使用来自 `GNU project` 的 `libiconv` 项目转码 `UTF-8` 输入，`C++20` 标准编译项目。

文件模块介绍



详细设计

bmp.hpp

提供 BMP 文件头信息结构体，一并给出第一步生成头文件信息结构体的函数。

```
1 namespace bmp{
2     using i32=int32_t;//LONG
3     using u32=uint32_t;//DWORD
4     using i16=int16_t;//
5     using u16=uint16_t;//WORD
6     #pragma pack(push,1)
7     //取消内存对齐，直接可以dump到文件
8     struct BMP_HEADER{
9         u16 bfType{0x4D42};
10        u32 bfSize;//
11        u16 bfReserved1{0};
12        u16 bfReserved2{0};
13        u32 bfOffBits;
14    };
```

```

15     struct BMP_INFO_HEADER{
16         u32 biSize{40};
17         i32 biWidth;
18         i32 biHeight;
19         u16 biPlanes{1};
20         u16 biBitCount;
21         u32 biCompression{0};
22         u32 biSizeImage;
23         i32 biXPelsPerMeter{11811},biYPelsPerMeter{11811};
24         u32 biClrUsed;//
25         u32 biClrImportant{0};
26     };
27     struct BMP_FILE_HEADER{
28         BMP_HEADER h1;
29         BMP_INFO_HEADER h2;
30     };
31     #pragma pack(pop)
32     inline BMP_FILE_HEADER make_rgb_header(i32 n,i32 m){
33         BMP_FILE_HEADER a;
34         tie(a.h2.biHeight,a.h2.biWidth)=make_tuple(n,m);
35         a.h1.bfOffBits=sizeof(a.h1)+sizeof(a.h2);
36         return a;
37     }
38
39 }

```

image.hpp

提供抽象图像信息模板类原型 `_bitmap<T>`，并特化为 `bitmap` 和 `rgbmap`。其中 `bitmap` 将在读取字库的 `fonts.hpp` 中用到，`rgbmap` 则为最终输出 BMP 文件用到。然后实现 `rgbmap` 的转 `bmp` 文件字符串函数。

```

1  using rgb=array<uint8_t,3>;
2
3  template<class T>
4  struct _bitmap{
5      size_t n,m;
6      vec<T> c;
7      inline _bitmap()=default;
8      inline _bitmap(size_t n,size_t m):n(n),m(m){
9          c.resize(n*m);
10     }
11     inline _bitmap(size_t n,size_t m,T a):n(n),m(m){
12         c.resize(n*m,a);
13     }
14     inline decltype(auto) operator()(size_t i,size_t j){//使用函数调用重载实现
        二维数组访问
        return c[i*m+j];
15     }
16     inline decltype(auto) operator()(size_t i,size_t j)const{
        return c[i*m+j];
17     }
18     inline string to_bmp();
19
20

```

```

21 };
22 template<>
23 inline string _bitmap<rgb>::to_bmp(){// 实现将rgbmap转bmp文件数据
24     set<rgb> s;
25     string str;
26     auto h=bmp::make_rgb_header(n,m);
27     for (int i=n-1;i>=0;i--)
28         for (int j=0;j<m;j++)
29             s.insert((*this)(i,j));
30     h.h1.bfSize=str.size()+sizeof(h);// 补充文件头信息
31     h.h2.biClrUsed=s.size();
32     h.h2.biSizeImage=0;
33     h.h2.biBitCount=24;
34     dump_to_string(h,str);//
35     for (int i=n-1;i>=0;i--)
36         for (int j=0;j<m;j++)
37             dump_to_string((*this)(i,j),str);
38     return str;
39 }
40
41 using bitmap=_bitmap<bool>;
42 using rgbmap=_bitmap<rgb>;

```

fonts.hpp

传入LiShu56.txt接受的 GB 系列中文单字编码，返回点阵文字信息的bitmap。

```

1  #pragma once
2  #include "cardinal.hpp"
3  #include "image.hpp"
4  inline optional<bitmap> find_bitmap(string hexcode){
5      string header="CurCode: "+hexcode,cc;
6      ifstream is("LiShu56.txt");
7      while ((getline(is,cc),is)&&!cc.starts_with(header));
8      if (cc.starts_with(header)){
9          bitmap b;
10         getline(is,cc);
11         int cntm=0,cntn=0;
12         while ((getline(is,cc),is)&&(cntm=[&]() {
13             int cnt=0;
14             for (auto i:cc){
15                 if (i=='X')
16                     b.c.push_back(1),cnt++;
17                 if (i=='_')
18                     b.c.push_back(0),cnt++;
19             }
20             return cnt;
21         }())){
22             b.m=cntm;
23             cntn++;
24         }
25         b.n=cntn;
26         return b;
27     }

```

```
28     return {};  
29 }
```

conv.hpp

简单封装libiconv，没处理错误码。

```
1  #pragma once  
2  #include "cardinal.hpp"  
3  #include "iconv.h"  
4  class converter{  
5      iconv_t bg;  
6      public:  
7      inline converter(const string& to,const string& from){  
8          bg=iconv_open(to.c_str(),from.c_str());  
9      }  
10     inline string operator()(const string& str){  
11         string w;w.resize(str.size());  
12         size_t wleft=w.size(),sleft=str.size();  
13         char *wc=const_cast<char*>(w.c_str());  
14         char* sc=const_cast<char*>(str.c_str());  
15         iconv(bg,&sc,&sleft,&wc,&wleft);  
16         w.resize(wc-w.c_str());  
17         return w;  
18     }  
19     inline ~converter(){  
20         iconv_close(bg);  
21     }  
22 };
```

main.cpp

具体实现代码。

```
1  string name,filename="out.bmp";  
2  string dump_to_hex(uint8_t c){//将一个unsigned char转成两位十六进制字符串  
3      auto single_to_hex=[](uint8_t a)->char{  
4          return a<10?a+'0':(a-10+'a');  
5      };  
6      return string()+single_to_hex(c/16)+single_to_hex(c%16);  
7  }  
8  
9  template<class T>  
10 string dump_to_hex(const T& a){//将一个类型成两位十六进制字符串  
11     string w;  
12     auto c=reinterpret_cast<uint8_t*>(&a);  
13     for (size_t i=0;i<sizeof(a);i++)  
14         w+=dump_to_hex(c[i]);  
15     return w;  
16 }  
17 string dump_to_hex(void* a,size_t n){//上一个函数的C风格版本
```



```

18     string w;
19     uint8_t* c=(uint8_t*)a;
20     for (size_t i=0;i<n;i++)
21         w+=dump_to_hex(c[i]);
22     return w;
23 }
24
25 void help(){//输出帮助信息
26     println("使用: $seal [options] [name]");
27     println("options:");
28     println("    -o [filename]      输出到filename");
29     println("    -h                打印此帮助信息");
30 }
31
32 void parse(vec<string> args){//简单解析命令行参数
33 #define byebye(exitcode) {help();exit(exitcode);}
34     array<bool,2> ba={};
35     if (args.size()==1) byebye(0);
36     for (size_t i=1;i<args.size();i++){
37         if (args[i]=="-o"){
38             if (i+1==args.size()||ba[0]) byebye(-1);
39             filename=args[i+1];
40             i++;
41             ba[0]=1;
42         }else if (args[i]=="-h"||args[i]=="--help"){
43             byebye(0);
44         }else{
45             if (ba[1]) byebye(-1);
46             name=args[i];
47             ba[1]=1;
48         }
49     }
50     return ;
51 #undef byebye
52 }
53
54 constexpr auto REDW=20;//输出印章红色边缘宽度
55 constexpr auto WITW=10;//输出印章红色边缘离文字宽度
56 constexpr auto BLK1W=REDW+WITW;
57 constexpr auto BLK2W=BLK1W*2;
58 constexpr auto RED=rgb{0,0,255};//红色的rgb值, rgb结构体的三个分别代表{b,g,r}
59 constexpr auto WIT=rgb{255,255,255};//白色
60
61 int main(int argc, char** argv){
62     parse({argv,argv+argc});//解析命令行参数
63     string buff;
64     buff=converter("GB18030","UTF-8")(name);//转码为字库的GB码
65     string hexbuff[2][2];
66     for (size_t i=0;i<8;i+=2)
67         hexbuff[0][i>>1]=dump_to_hex(&buff[i],2);//转成十六进制码
68
69     bitmap bmbuff[2][2];int pn=0,pm=0;
70
71     for (int i=0;i<2;i++){
72         for (int j=0;j<2;j++){
73             auto w=find_bitmap(hexbuff[i][j]);//查表

```

```

74         if (w){
75             pn=w->n;
76             pm=w->m;
77             bmbuff[i][j]=move(*w);
78         }
79     }
80 }
81
82 for (int i=0;i<2;i++)
83     for (int j=0;j<2;j++)
84         bmbuff[i][j].c.resize(pn*pm);
85
86 swap(bmbuff[0][0],bmbuff[0][1]);
87 swap(bmbuff[1][0],bmbuff[1][1]);
88 swap(bmbuff[0][0],bmbuff[1][1]); // 交换文字顺序使其符合印章顺序
89 rgbmap a(pn*2+BLK2W,pm*2+BLK2W,WIT); // 建立要输出的图像
90 for (int i=0;i<2;i++) // 填字
91     for (int j=0;j<2;j++){
92         int bi=BLK1W+i*pn,bj=BLK1W+j*pm;
93         for (int ii=0;ii<pn;ii++)
94             for (int jj=0;jj<pm;jj++)
95                 a(bi+ii,bj+jj)=bmbuff[i][j](ii,jj)?RED:WIT;
96     }
97 for (int i=0;i<REDW;i++) for (int j=0;j<a.m;j++) a(i,j)=RED; // 外圈涂红
98 for (int i=0;i<a.n;i++) for (int j=0;j<REDW;j++) a(i,j)=RED;
99 for (int i=0;i<REDW;i++) for (int j=0;j<a.m;j++) a(a.n-i-1,j)=RED;
100 for (int i=0;i<a.n;i++) for (int j=0;j<REDW;j++) a(i,a.m-j-1)=RED;
101 ofstream(filename,ios::binary)<<a.to_bmp(); // 转成bmp输出
102 return 0;
103 }

```

xmake.lua

xmake 读取的构建文件, 因为使用了 `iconv` 库所以需要额外链接 `iconv` 库, 即添加 `-liconv` 编译选项。

```

1  add_rules("mode.debug", "mode.release")
2
3  target("seal")
4      set_languages("c++20")
5      set_optimize("fastest")
6      add_includedirs("../headers")
7      add_includedirs("include")
8      set_kind("binary")
9      add_links("iconv")
10     add_files("src/*.cpp")

```

遇到的问题与解决方法

在现代 (Windows 需要开启实验性 Unicode 支持, Linux 需要指定 `LOCAL` 为 `zh_CN.UTF-8`) 操作系统中终端一般采用 `UTF-8` 编码, 遇到 GB 系列的编码需要进行转换, 使用来自 [GNU project](#) 的 `iconv` 库

就能很好的解决这个问题。

结束语

学习到了 C++ 工程开发的库链接技术。