五十音測驗

一、設計動機:

五十音是日文初學者遇到的第一個難題，如果只考慮清音的話，平假名和片假名加起來總共有 92 個字要記住，這時如果能有個測驗的小程式，應該會幫助不少。

二、輸入與輸出:

A.輸入:模式 mode、測驗題數 problem_n、答案 ans、是否再次測驗 end_or_conti
B.輸出:題目、錯題正解、答對率
C.範例:

1.平假名

```
請選擇模式，只要平假名輸入1，只要片假名輸入2，兩者都要輸入3
1
請輸入要測驗多少題？
5
第 1題:て？
yo
第 2題:ほ？
ha
第 3題:こ？
ko
第 4題:あ？
a
第 5題:ひ？
hi
答錯的題目有
第 1題:て應為te
第 2題:ほ應為ho
答對題數為 3，答對率為60.00%
是否要再進行一次，輸入0停止，輸入1繼續
```

2.片假名

```
請選擇模式，只要平假名輸入1，只要片假名輸入2，兩者都要輸入3
2
請輸入要測驗多少題？
5
第 1題:ホ？
ho
第 2題:チ？
chi
第 3題:ワ？
hu
第 4題:エ？
e
第 5題:サ？
sa
答錯的題目有
第 3題:ワ應為wa
答對題數為 4，答對率為80.00%
```

3.混合

```
請選擇模式，只要平假名輸入1，只要片假名輸入2，兩者都要輸入3
3
請輸入要測驗多少題？
10
第 1題:マ？
ma
第 2題:ケ？
ke
第 3題:コ？
ko
第 4題:ス？
ne
第 5題:ラ？
su
第 6題:な？
na
第 7題:も？
mo
第 8題:ヒ？
hi
第 9題:ろ？
so
第10題:ち？
chi
答錯的題目有
第 4題:ス應為su
第 5題:ラ應為ra
第 9題:ろ應為ro
答對題數為 7，答對率為70.00%
是否要再進行一次，輸入0停止，輸入1繼續
```

三、運作方式與步驟:

1.先請使用者輸入要進行的模式(1.平假名 2.片假名 3.混合)、題數、打亂題目順序
2.作答、輸出錯題的正確答案、答對題數與答對率

四、程式碼解說:

最外層迴圈:

```
25      bool end_or_conti;
26      do{
```

```
141         cout<<"是否要再進行一次，輸入0停止，輸入1繼續"<<endl;
142         cin>>end_or_conti;
143     }while(end_or_conti);
```

使用 do-while 讓使用者先進行一次測驗，待測驗結束再詢問使用者是否再次進行。

步驟 1：輸入模式

```
27          cout<<"請選擇模式，只要平假名輸入1，只要片假名輸入2，兩者都要輸入3"<<endl;
28          int mode;
29          cin>>mode;
30          while(mode!=1&&mode!=2&&mode!=3){
31              cout<<"請重新輸入有效數字，只要平假名輸入1，只要片假名輸入2，兩者都要輸入3"<<endl;
32              cin>>mode;
33          }
```

輸入 mode(1 代表平假名、2 代表片假名、3 代表混合)，並使用 while 判斷是否為 1、2、3 任一數字，若為否則進入 while 迴圈請使用者重新輸入。

步驟 1：輸入題數

```
35          cout<<"請輸入要測驗多少題?"<<endl;
36          int problem_n;
37          cin>>problem_n;
38
39          vector <int> problem;
40          for(int i=0;i<46;i++){
41              problem.push_back(i);
42          }
43
44          int correct_n=0;
45          vector <int> wrong;
46
47          random_order(&problem,problem.size());
```

1.輸入 promblem_n 代表題數
2.同時建立一個有 46 個元素(因五十音實際上只有 46 個，也就是題數最多只能有 46 題)的陣列 problem，其中元素為 1~46
3.為了之後要計算答對率，宣告一個變數 correct_n 來計答對題數
4.宣告一個名為 wrong 的向量，用來計答錯哪些題
5.random_order 用來打亂題目，也就是 1~46 的順序

步驟 1：打亂題目順序

```
14  void random_order(vector <int>* a,int n){
15      srand(time(0));
16      for(int i=0;i<n;i++){
17          int x=rand()%n;
18          int temp=a->at(i);
19          a->at(i)=a->at(x);
20          a->at(x)=temp;
21      }
22  }
```

這邊將打亂題目寫成獨立一個函數，因為沒有回傳值，所以宣告為 void 型別。而這個函數有兩個引數，一個為指標向量 a，另一個數字 n 為向量的大小。首先使用 srand(time(0))來初始化 rand()，接著進入一個 for 迴圈，每一次操作都是取第 i 個和第 x 個指標所指向的元素值交換，其中 x 為 0~45 之間任一整數。

步驟 2 : mode=1(平假名)

```
50              case 1:{
51                  for(int i=0;i<problem_n;i++){
52                      string ans;
53                      cout<<"第"<<setw(2)<<i+1<<"題:"<<hiragana[problem[i]]<<"?"<<endl;
54                      cin>>ans;
55                      int ans_pos;
56                      for(int j=0;j<46;j++){
57                          if(ans==spelling[j]){
58                              ans_pos=j;
59                              break;
60                          }
61                      }
62                      if(ans_pos==problem[i]){
63                          correct_n+=1;
64                      }else{
65                          wrong.push_back(i);
66                      }
67                  }
68                  cout<<"答錯的題目有"<<endl;
69                  for(int i=0;i<wrong.size();i++){
70                      cout<<"第"<<setw(2)<<wrong[i]+1<<"題:"<<hiragana[problem[wrong[i]]]
71                          <<"應為"<<spelling[problem[wrong[i]]]<<endl;
72                  }
73                  break;
74              }
```

這邊使用 switch 來進行不同情形的處理

先進入一個 for 迴圈，所執行的次數為所要測驗的題數，每一次的執行內容為:

假設現在為第 i 次的迴圈

1.首先先輸出題目 hiragana[problem[i]]，其中 hiragana[]是一個建立在主函數之外的唯讀 string array，裡面有 46 個平假名，並且按照順序排列，problem[i]則是 0~45 之間的隨機整數。

2.接著將作答的答案存入一個名為 ans 的 string，並利用 for 迴圈一個個比對 ans 所對應 spelling[]中的元素，最後將 ans 在 spelling[]中所對應到的編號存入 ans_pos。這裡的 spelling[]和 hiragana[]一樣，是建立在主函數之外的唯讀 string array，用來儲存每個平假名或片假名的英文拼音，順序也和兩者一樣，例如 hiragana[5]是「か」，而 spelling[5]也對應到「ka」。

3.因為 hiragana[]和 spelling[]所對應的字是相同的，所以只要檢驗他們的元素編號是否相同即可。另外在第 i 次下的出現的平假名編號為 problem[i]，而輸入答案的編號則是 ans_pos。

4.如果編號相同，則將 correct_n 加一代表達對，如果答錯，則將答錯的題號 i 存入 wrong 裡。

5.輸出錯誤的題目，使用 for 迴圈來一次次輸出錯題的正解。

步驟 2 : mode=2(片假名)

```
75              case 2:{
76                  for(int i=0;i<problem_n;i++){
77                      string ans;
78                      cout<<"第"<<setw(2)<<i+1<<"題:"<<katakana[problem[i]]<<"?"<<endl;
79                      cin>>ans;
80                      int ans_pos;
81                      for(int j=0;j<46;j++){
82                          if(ans==spelling[j]){
83                              ans_pos=j;
84                              break;
85                          }
86                      }
87                      if(ans_pos==problem[i]){
88                          correct_n+=1;
89                      }else{
90                          wrong.push_back(i);
91                      }
92                  }
93                  cout<<"答錯的題目有"<<endl;
94                  for(int i=0;i<wrong.size();i++){
95                      cout<<"第"<<setw(2)<<wrong[i]+1<<"題:"<<katakana[problem[wrong[i]]]
96                          <<"應為"<<spelling[problem[wrong[i]]]<<endl;
97                  }
98                  break;
99              }
```

主要與平假名相同，只是將 hiragana[]改為 katakana[]，且 katakana[]的字母順序也和 hiragana[]相同

步驟 2 : mode=3(混合)

```
100             case 3:{
101                 int state;
102                 vector <int> wrong_state;
103                 for(int i=0;i<problem_n;i++){
104                     string ans;
105                     state=rand()%2;
106                     if(state==0){
107                         cout<<"第"<<setw(2)<<i+1<<"題:"<<hiragana[problem[i]]<<"?"<<endl;
108                     }else{
109                         cout<<"第"<<setw(2)<<i+1<<"題:"<<katakana[problem[i]]<<"?"<<endl;
110                     }
111                     cin>>ans;
112                     int ans_pos;
113                     for(int j=0;j<46;j++){
114                         if(ans==spelling[j]){
115                             ans_pos=j;
116                             break;
117                         }
118                     }
119                     if(ans_pos==problem[i]){
120                         correct_n+=1;
121                     }else{
122                         wrong.push_back(i);
123                         wrong_state.push_back(state);
124                     }
125                 }
126                 cout<<"答錯的題目有"<<endl;
127                 for(int i=0;i<wrong.size();i++){
128                     cout<<"第"<<setw(2)<<wrong[i]+1<<"題:";
129                     if(wrong_state[i]==0){
130                         cout<<hiragana[problem[wrong[i]]];
131                     }else{
132                         cout<<katakana[problem[wrong[i]]];
133                     }
134                     cout<<"應為"<<spelling[problem[wrong[i]]]<<endl;
135                 }
136                 break;
137             }
```

最開始先宣告一個變數 state 用來處理要輸出平假名 0 或片假名 1，因為要隨機出題，所以將 rand()%2 所得的值給 state。接下來大部分的處理與上兩種 case 相同，但遇到在 case 1 是 hiragana[] 的部分就改成用 if 來判斷目前 state 的值，如果是 0 就以 hiragana[]作為輸入輸出，反之遇到 1 就以 katakana[]作為輸入輸出

步驟 2 : 輸出答對題數與答對率

```
139             cout<<"答對題數為"<<setw(2)<<correct_n<<"，答對率為"<<setw(5)<<fixed
140                 <<setprecision(2)<<double(correct_n)/double(problem_n)*100<<"%"<<endl;
```

因為這部分在三種模式都會執行且處理方法一樣，所以在走完 switch 後再來處理即可。其中在這邊答對題數 correct_n 和題數 problem_n 都必須進行一次型別轉換，否則無法算出答對率。