Project 2： Simple ADIF Database

1 Project Requirements

The project needed to implement a C++ program that could import ADIF or CSV format ADIF data, store it in its own database file in binary format, and export it as an ADIF or CSV format file. Use command line arguments to specify what to do and the file name to import/export. Here are the command-line arguments that the program implements:

- `-i <file name>`: Import ADIF or CSV files, depending on the file name suffix to determine the type of file to import (`.adi` or `.csv`). The imported data is used to update the binary database file. The records with the same primary key is considered to be the same record. If there are multiple records with the same primary key in the import file, the last appearing data is the one that prevails. When importing data, records whose primary key is not the same as any of the records in the database are added as new data, while records whose primary key already exists in the database are used to update the existing data.
- `-o <file name>`: Export ADIF or CSV files, depending on the file name suffix to determine the type of file to export.
- `-s <call>`: Search for records with the exact same `call` field according to the command line argument and output them to the command line. The format is explained below
- `-l <start time> <end time>`: The time is expressed according to `YYYYMMDDhhmmss`. Output all records within the closed interval of this time to the command line. The format is explained below.
- `-d <file name>`: Import ADIF or CSV files, depending on the file name suffix to determine the type of file to import (`.adi` or `.csv`). Only the primary key must be imported. In the records in the binary database file, remove the records whose primary key exists in the imported file. Your program can first export the data in the binary database file as text, perform string matching and deletion, and then overwrite the original binary database file.

2 Main ideas and thoughts

2.1 operation" -i "

When performing the -i operation, the idea is that the file to be imported will be processed briefly so that it can be used later. So I processed both files into the same format and then put `database.bin` file. At the same time, the same primary key record is overwritten, file letter case conversion,.adi file header is deleted, records are arranged in descending order by primary key and other effects are realized in this operation

- Same primary key record overlay and primary key descending order implementation: I use a map container to store primary keys and other contents to achieve descending order. After reading a new file, in order to save the contents of the original map, I read `database.bin` first and combine it with the newly imported file into the temporary file `temp_database.bin`. Then write the contents of `temp_database.bin` back to `database.bin` file to achieve the primary key record overwrite
- Implementation of case conversion of file letters: converts all lowercase letters to uppercase letters when importing files
- .adi file header deletion implementation: when importing.adi file ignore all contents before `<EOH>`, if no `<EOH>` is found, all files are imported

The format of the final import is shown below:

```
20070908102100<FREQ>18080<MODE>SSB<CALL>RV9LF<RST_RCVD>559<RST_SENT>579
20070908094800<FREQ>14215<MODE>SSB<CALL>RK9UE<RST_RCVD>59<RST_SENT>59

```

2.2 operation" -o"

When making - o operations, my idea is to use the map container respectively to store the field names and field content will field according to the dictionary sequence alignment.Then determine the output file format according to the input file name

The format of the final import is shown below:

```
<QSO_DATE:8>20070908<TIME_ON:6>102100<CALL:5>RV9LF<FREQ:5>18080<MODE:3>SSB<RST_RCVD:3>559<RST_SENT:3>579<EOR>//.adi
QSO_DATE,TIME_ON,APP_WRITELOG_COUNTRY,APP_WRITELOG_MULS,APP_WRITELOG_P,APP_WRITELOG_PREF,BAND,CALL,FREQ,MODE,PFX,RST_RCVD,RST_SENT,SRX,STX
20070908,102100,,,,,,RV9LF,18080,SSB,,559,579,,//.csv
```

2.3 operation" -s"

When making - s operations, my idea is to open the `database.bin` file, look for the contents of the `<call>` field in each record, and output the record if it is the same as what you are looking for

2.4 operation" -l"

When making -l operations, my idea is to open the `database.bin` file,Determine whether the time of each line is within the input time range, and output if it is

2.5 operation" -d"

When making -l operations, my idea is to open the record file to be deleted and place the primary key to be deleted in the set container. Empty the temporary file `temp_database.bin` and open the `database.bin` file to determine whether each line of records is in the set container to be deleted. Import the modified database.bin into `temp_database.bin` . Then import `temp_database.bin` into `database.bin`

3 How the program works

3.1 Input

- The program passes the command line as an argument to the main function `int argc, char* argv[]`
- In the batch file, first compile the program into an executable file with `g++ code.cpp -o code.exe` , after entering parameters "-i", "-o", etc

3.2 Output

- After running the batch file, the program will output the input command in the command line or export the file

3.3 Mode of operation

- Compile the program on the command line and enter the required instructions,like:

```
g++ code.cpp -o code.exe
code.exe -i a.adi
```

- Or just double-click the batch file

4 Explanation of important parts of the source code

- function `main` :According to the input parameter to determine the type of instruction, enter the different instruction function
- function `convert` :Converts the middle character of "< >" in the passed string to uppercase, leaving all other characters unchanged. This ensures that the field names are not case sensitive and that the output is uppercase and the field content is case sensitive
- function `import_file` :Import the corresponding file. Open the file based on the file name
- function `export_file` :The corresponding file export operation imports data in the database to the corresponding file according to the file name passed in
- function `search_records` :Corresponding to the search data operation, according to the string passed in the database field and the string is the same as the output string
- function `output_records_in_time_range` :Operations that output data in the specified time range. Searches the data in the corresponding time range in the database according to the incoming time range and outputs the data
- function `delete_records` :Corresponding to the operation of deleting data, open the corresponding file according to the incoming file name, search for the data in the database that is gradually the same as the data in the corresponding file and delete it

5 Testing result

Testing code:Perform the following operations in sequence

- Import the a.adi, b.adi, and c.csv files
- Search data for BD8GK
- Output the data between 20070130050400 and 20070131113100
- Output the files in the database as test.adi and test.csv
- Delete b.adi
- Output the file in the database as test.csv

```
@echo off
:: compile the program
g++ code.cpp -o code.exe
:: display the C++ version
echo Using C++ version: c++ 11
```

```
code.exe -i a.adi
code.exe -i b.adi
code.exe -i c.csv
code.exe -s BD8GK
code.exe -l 20070130050400 20070131113100
code.exe -o test.adi
code.exe -o test.csv
code.exe -d b.adi
code.exe -o test.csv
pause
```

Testing result:

- test.adi

```
<QSO_DATE:8>20070908<TIME_ON:6>102100<CALL:5>RV9LF<FREQ:5>18080<MODE:3>SSB<RST_RCVD:3>559 <RST_SENT:3>579<EOR>
<QSO_DATE:8>20070908<TIME_ON:6>100100<CALL:6>E20WXA<FREQ:12>"14220,123456" <MODE:3>SSB<RST_RCVD:2>59<RST_SENT:2>59<EOR>
<QSO_DATE:8>20070908<TIME_ON:6>094800<CALL:5>RK9UE<FREQ:5>14215<MODE:3>SSB<RST_RCVD:2>59< RST_SENT:2>59<EOR>
<QSO_DATE:8>20070526<TIME_ON:6>011234<APP_WRITELOG_COUNTRY:5>Japan <APP_WRITELOG_P:1>1 <APP_WRITELOG_PREF:2>JA <BAND:3>15m <CALL:6>JA1MVK <FREQ:6>21.013 <MODE:2>CW <PFX:3>JA1 <RST_RCVD:3>599 <RST_SENT:3>599 <SRX:3>024 <STX:1>6 <EOR>
<QSO_DATE:8>20070526<TIME_ON:6>011106<APP_WRITELOG_COUNTRY:5>Japan <APP_WRITELOG_MULS:1>4 <APP_WRITELOG_P:1>1 <APP_WRITELOG_PREF:2>JA <BAND:3>15m <CALL:6>JP1QDH <FREQ:6>21.013 <MODE:2>CW <PFX:3>JP1 <RST_RCVD:3>599 <RST_SENT:3>599 <SRX:3>025 <STX:1>5 <EOR>
<QSO_DATE:8>20070526<TIME_ON:6>011037<APP_WRITELOG_COUNTRY:5>Japan <APP_WRITELOG_MULS:1>3 <APP_WRITELOG_P:1>1 <APP_WRITELOG_PREF:2>JA <BAND:3>15m <CALL:6>JR1NHD <FREQ:6>21.013 <MODE:2>CW <PFX:3>JR1 <RST_RCVD:3>599 <RST_SENT:3>599 <SRX:3>009 <STX:1>4 <EOR>
<QSO_DATE:8>20070526<TIME_ON:6>010934<APP_WRITELOG_COUNTRY:5>Japan <APP_WRITELOG_P:1>1 <APP_WRITELOG_PREF:2>JA <BAND:3>15m <CALL:6>JA1PJS <FREQ:6>21.013 <MODE:2>CW <PFX:3>JA1 <RST_RCVD:3>599 <RST_SENT:3>599 <SRX:3>003 <STX:1>3 <EOR>
<QSO_DATE:8>20070526<TIME_ON:6>010813<APP_WRITELOG_COUNTRY:5>Japan <APP_WRITELOG_MULS:1>2 <APP_WRITELOG_P:1>1 <APP_WRITELOG_PREF:2>JA <BAND:3>15m <CALL:6>JA1MCU <FREQ:6>21.013 <MODE:2>CW <PFX:3>JA1 <RST_RCVD:3>599 <RST_SENT:3>599 <SRX:3>022 <STX:1>2 <EOR>
<QSO_DATE:8>20070526<TIME_ON:6>010631<APP_WRITELOG_COUNTRY:5>Japan <APP_WRITELOG_MULS:1>1 <APP_WRITELOG_P:1>1 <APP_WRITELOG_PREF:2>JA <BAND:3>15m <CALL:8>JH1GUO/4 <FREQ:6>21.013 <MODE:2>CW <PFX:3>JH4 <RST_RCVD:3>599 <RST_SENT:3>599 <SRX:3>024 <STX:1>1 <EOR>
<QSO_DATE:8>20070203<TIME_ON:6>102100<CALL:5>rv9LF <FREQ:5>18080 <MODE:3>SSB <RST_RCVD:3>559 <RST_SENT:3>579 <EOR>
<QSO_DATE:8>20070203<TIME_ON:6>100100<CALL:6>E20wxA <FREQ:5>14220 <MODE:3>SSB <RST_RCVD:2>59 <RST_SENT:2>59 <EOR>
<QSO_DATE:8>20070203<TIME_ON:6>094800<CALL:5>RK9UE <FREQ:5>14215 <MODE:3>SSB <RST_RCVD:2>59 <RST_SENT:2>59 <EOR>
```

```
<QSO_DATE:8>20070203<TIME_ON:6>094200<CALL:8>BG8ATI/8 <FREQ:5>14260 <MODE:3>SSB
<RST_RCVD:2>59 <RST_SENT:2>59 <EOR>
<QSO_DATE:8>20070203<TIME_ON:6>093100<CALL:5>Ua9ZZ <FREQ:5>18078 <MODE:2>CW
<RST_RCVD:3>559 <RST_SENT:3>559 <EOR>
<QSO_DATE:8>20070203<TIME_ON:6>071400<CALL:6>XU7XRO <FREQ:5>18077 <MODE:2>CW
<RST_RCVD:3>599 <RST_SENT:3>599 <EOR>
<QSO_DATE:8>20070203<TIME_ON:6>071200<CALL:5>BX2AK <FREQ:5>14018 <MODE:2>CW
<RST_RCVD:3>599 <RST_SENT:3>599 <EOR>
<QSO_DATE:8>20070131<TIME_ON:6>102800<CALL:6>VR2VAC <FREQ:5>14180 <MODE:3>SSB
<RST_RCVD:2>59 <RST_SENT:2>59 <EOR>
<QSO_DATE:8>20070130<TIME_ON:6>113100<CALL:5>RA9MP <FREQ:5>14021 <MODE:2>CW
<RST_RCVD:3>599 <RST_SENT:3>599 <EOR>
<QSO_DATE:8>20070130<TIME_ON:6>050400<CALL:5>BD8GK <FREQ:5>14270 <MODE:3>SSB
<RST_RCVD:2>59 <RST_SENT:2>59 <EOR>
```

- test.csv

```
QSO_DATE,TIME_ON,CALL,FREQ,MODE,RST_RCVD,RST_SENT
20070908,102100,RV9LF,18080,SSB,559,579
20070908,100100,E20WXA,"14220,123456",SSB,59,59
20070908,094800,RK9UE,14215,SSB,59,59
20070203,102100,rv9LF ,18080 ,SSB ,559 ,579
20070203,100100,E20wxA ,14220 ,SSB ,59 ,59
20070203,094800,RK9UE ,14215 ,SSB ,59 ,59
20070203,094200,BG8ATI/8,14260,SSB,59,59
20070203,093100,UA9ZZ,18078,CW,559,559
20070203,071400,XU7XRO ,18077 ,CW ,599 ,599
20070203,071200,BX2AK ,14018 ,CW ,599 ,599
20070131,102800,VR2VAC ,14180 ,SSB ,59 ,59
20070130,113100,RA9MP ,14021 ,CW ,599 ,599
20070130,050400,BD8GK ,14270 ,SSB ,59 ,59
```

- Search data for BD8GK

```
QSO_DATE:20070130,TIME_ON:050400 ,FREQ:14270 ,MODE:SSB ,CALL:BD8GK ,RST_RCVD:59 ,RST_SENT:59

1 record(s) found
```

- Output the data between 20070130050400 and 20070131113100

```
QSO_DATE:20070131,TIME_ON:102800 ,FREQ:14180 ,MODE:SSB ,CALL:VR2VAC ,RST_RCVD:59 ,RST_SENT:59

QSO_DATE:20070130,TIME_ON:113100 ,FREQ:14021 ,MODE:CW ,CALL:RA9MP ,RST_RCVD:599 ,RST_SENT:599

QSO_DATE:20070130,TIME_ON:050400 ,FREQ:14270 ,MODE:SSB ,CALL:BD8GK ,RST_RCVD:59 ,RST_SENT:59

3 record(s) found
```

6 Discussions

- For the.adi file without , I initially thought that the import should not be carried out, and I understood all the contents of the file that should be imported under the teaching assistant's explanation
- When I imported files continuously, I found that the last data would be emptied each time. After discussion with my classmates, I reset the method of writing files to solve the problem
- Since the new data may overwrite the old data, the data structure of the last run of the program should be retained when importing. After discussion with my classmates, I decided to set up a temporary file to save the data