概念题

1. C++中输入/输出（I/O）分成几类？分别是什么？
   三类. 面向控制台的 I/O，面向文件的 I/O 和面向字符串变量的 I/O.

2. 请简述 C++中流式文件的概念
   在 C++中，把文件看成是由一系列字节所构成的字节串
   对文件中数据的操作(输入/输出)逐个字节顺序进行，称为流式文件。

3. 请简述 C++中对文件数据进行读写的过程
   打开文件：把程序内部表示文件的变量/对象与外部具体文件关联，创建内存缓冲区
   文件读/写：存取文件中的内容。
   关闭文件：把暂存在缓冲区中的内容写入到文件，归还打开文件时申请的内存资源

编程题

1. 以表格形式输出当 x = 1°，2°，…，10°时 sin(x)、cos(x)和 tan(x)的值。
   输出时，三角函数值的宽度为 10，左对齐，保留小数点后 5 位，如图所示。

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int main()
{
    double rad_to_dag = 45.0 / atan(1.0);
    cout << setiosflags(ios::left) << "x\tsin(x)\t\tcos(x)\t\ttan(x)" << endl;
    for (int i = 1; i <= 10; i++) {
        double dag = (double)i / rad_to_dag;
        cout << i << '\t' << setiosflags(ios::fixed) << setprecision(5) <<
            sin(dag) << "\t\t" << cos(dag) << "\t\t" << tan(dag) << endl;
    }
    return 0;
}
```

2. 二进制文件“A.dat"和”B.dat“分别保存两个钟点工一天中可被安排工作的所有时间段。
   二进制文件”C.dat“是按时间顺序保存着两人一天中能被安排到一起工作的所有时间段。
   用控制台分别输入两工人可工作的所有时间段，并各自保存到“A.dat"和”B.dat“；
   读取“A.dat"和”B.dat“并生成”C.dat“；读取”C.dat“并用控制台输出。
   <注：时间段简化为整点时间段，例如：8:00~10:00,13:00~16:00>

```
#include <iostream>
#include <fstream>
```

```cpp
#include <algorithm>
using namespace std;

int main() {
    cout << "工作时段始末为整数，以空格分隔，换行分隔时段，输入E结束." << endl;
    char temp[10];
    char num[5];
    cout << "请输入第一个工人可工作的所有时段:" << endl;
    ofstream out_A("A.dat", ios::out | ios::binary);
    if (!out_A) exit(-1);
    cin.getline(temp, 9);
    while (strcmp(temp, "E")) {
        int count = 0;
        for (int i = 0; temp[i] != '\0'; i++) {
            if (temp[i] == ' ') {
                num[count] = '\0';
                out_A << num << ' ';
                count = 0;
            }
            else {
                num[count] = temp[i];
                count += 1;
            }
        }
        num[count] = '\0';
        out_A << num << '\n';
        cin.getline(temp, 9);
    }
    out_A.close();
    cout << "请输入第二个工人可工作的所有时段:" << endl;
    ofstream out_B("B.dat", ios::out | ios::binary);
    if (!out_B) exit(-1);
    cin.getline(temp, 9);
    while (strcmp(temp, "E")) {
        int count = 0;
        for (int i = 0; temp[i] != '\0'; i++) {
            if (temp[i] == ' ') {
                num[count] = '\0';
                out_B << num << ' ';
                count = 0;
            }
            else {
                num[count] = temp[i];
                count += 1;
```

```
        }
    }
    num[count] = '\0';
    out_B << num << '\n';
    cin.getline(temp, 9);
}
out_B.close();
ifstream in_A("A.dat", ios::in | ios::binary);
ifstream in_B("B.dat", ios::in | ios::binary);
int A[10][2];
int B[10][2];
int C[10][2];
int a = 0, b = 0, c = 0;
while (!in_A.fail()) {
    in_A >> A[a][0] >> A[a][1];
    a += 1;
}
a -= 1;
in_A.close();
while (!in_B.fail()) {
    in_B >> B[b][0] >> B[b][1];
    b += 1;
}
b -= 1;
in_B.close();
for (int i = 0; i < a; i++) {
    for (int j = 0; j < b; j++) {
        // cout << A[i][0] << ' ' << A[i][1] << ' ' << B[j][0] << ' ' << B[j][1] <<endl;
        if (A[i][0] >= B[j][1] || A[i][1] <= B[j][0])
            continue;
        else {
            C[c][0] = max(A[i][0], B[j][0]);
            C[c][1] = min(A[i][1], B[j][1]);
            c += 1;
        }
    }
}
ofstream out_C("C.dat", ios::out | ios::binary);
for (int k = 0; k < c; k++)
    out_C << C[k][0] << ' ' << C[k][1] << '\n';
out_C.close();
char start[10], end[10];
ifstream in_C("C.dat", ios::in | ios::binary);
while (!in_C.fail()) {
```

```
        in_C >> start >> end;
        cout << start << ' ' << end << endl;
    }
    return 0;
}
```

3. 在某单位的文本文件“workers.txt”中，每行记录了一名员工（Worker）的信息：
编号，姓名，电话号码，邮政编号和住址。
通讯录系统（AddressBook）包含一系列对“workers.txt”的操作：
创建一个全新的文件“workers.txt”；
void add(Worker &worker)：往文件末尾添加一名员工的信息；
Worker search(string id)：在文件中查找并返回编号为 id 的员工；
void modify(Worker &worker)：修改该员工除了编号外的信息（位置不变）。
完成 Worker 类，重载操作符“<<”实现 Worker 类对象的输出；完成 Address 类。

```
#include <iostream>
#include <fstream>
using namespace std;

class Worker {
    string id;
    string name;
    string tel;
    string code;
    string address;
    friend class AddressBook;
public:
    Worker(string i, string n, string t, string c, string a) {
        id = i; name = n; tel = t; code = c; address = a;
    }
    friend ostream& operator << (ostream& out, const Worker& w);
};

ostream& operator << (ostream& out, const Worker& w) {
    out << w.id << ',' << w.name << ',' << w.tel << ',' << w.code << ',' << w.address;
    return out;
}

class AddressBook {
public:
    AddressBook() {
        ofstream out_Book("workers.txt", ios::out);
        if (!out_Book) exit(-1);
        out_Book.close();
```

```
}
void add(Worker& worker) {
    fstream f_Book("workers.txt", ios::in | ios::app);
    if (!f_Book) exit(-1);
    f_Book << worker.id << ' ' << worker.name << ' ' << worker.tel << ' '
        << worker.code << ' ' << worker.address << endl;
    f_Book.close();
}
Worker search(string id) {
    ifstream in_Book("workers.txt", ios::in);
    if (!in_Book) exit(-1);
    char buffer[100];
    string temp_id;
    string temp_name;
    string temp_tel;
    string temp_code;
    string temp_address;
    while (!in_Book.fail()) {
        in_Book >> temp_id;
        if (temp_id.compare(id) == 0) {
            in_Book >> temp_name >> temp_tel >> temp_code >> temp_address;
            Worker temp(temp_id, temp_name, temp_tel, temp_code, temp_address);
            in_Book.close();
            return temp;
        }
        else {
            in_Book.getline(buffer, 99);
        }
    }
    in_Book.close();
}
void modify(Worker& worker) {
    ifstream in_Book("workers.txt", ios::in);
    if (!in_Book) exit(-1);
    int line = 0;
    bool exist = false;
    char buffer[100];
    string temp_id;
    while (!in_Book.fail()) {
        in_Book >> temp_id;
        if (temp_id.compare(worker.id) == 0) {
            exist = true;
            break;
        }
```

```
            else {
                in_Book.getline(buffer, 99);
            }
            line += 1;
        }
        in_Book.close();
        if (exist) {
            fstream out_Book("workers.txt", ios::in | ios::out);
            if (!out_Book) exit(-1);
            char buffer[100];
            string temp_id;
            for (int i = 0; i < line; i++) {
                out_Book.getline(buffer, 99);
            }
            out_Book << worker.id << ' ' << worker.name << ' ' << worker.tel << ' '
                << worker.code << ' ' << worker.address << endl;
            out_Book.close();
        }
    }
};

int main()
{
    AddressBook BOOK;
    Worker Damon("123", "Damon", "123456", "100000", "Essex");
    Worker Graham("124", "Graham", "138709", "100005", "London");
    Worker Alex("145", "Alex", "577755", "200030", "Bristol");
    Worker Dave("101", "Dave", "883043", "000001", "Dublin");
    BOOK.add(Damon);
    BOOK.add(Graham);
    BOOK.add(Alex);
    BOOK.add(Dave);
    Worker Cheese = BOOK.search("145");
    cout << Cheese << endl;
    Worker Graham_Moved("124", "Graham", "138709", "900014", "Idaho");
    BOOK.modify(Graham_Moved);
    return 0;
}
```