高级程序设计 2020 春

实验报告

指导老师：陈家骏 黄书剑 姓名：王晨渊, 学号：181220057

1 概念题

1.1 C++ 中输入/输出（I/O）分成几类？分别是什么？

1.1.1 面向控制台的 I/O

从标准输入设备中获得数据，把程序结果从标准输出设备中输出。

1.1.2 面向文件的 I/O

从外存文件中获得数据，把程序结果保存到外存文件中。

1.1.3 面向字符串变量的 I/O

从程序中的字符串变量中获取数据，把程序结果保存到字符串变量中。

1.2 请简述 C++ 中流式文件的概念。

从手册上抄来的。A stream is an abstraction that represents a device on which input and ouput operations are performed. A stream can basically be represented as a source or destination of characters of indefinite length.

1.3 请简述 C++ 中对文件数据进行读写的过程。

创建对应的 fstream 类，保证打开成功后进行读写，读写完成后调用 close 方法。

2 编程题

2.1 表格

```
1  #include <iostream>
2  #include <iomanip>
3  #include <cmath>
4  using namespace std;
5  #define PI 3.141592653589
```

```
int main()
{
    cout.setf(ios::left);
    cout<<setw(3)<<"x";
    cout<<setw(10)<<"sin(x)"<<setw(10)<<"cos(x)"<<setw(10)<<"tan(x)"<<endl;
    for(int i=1;i<=10;i++)
    {
        cout<<setw(3)<<i;
        cout<<setw(10)<<setiosflags(ios::fixed)<<setprecision(5)<<sin((double)i/1
        cout<<setw(10)<<setiosflags(ios::fixed)<<setprecision(5)<<cos((double)i/1
        cout<<setw(10)<<setiosflags(ios::fixed)<<setprecision(5)<<tan((double)i/1
        cout<<endl;
    }
}
```

2.2 时间分配

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#include <regex>
#include <cstdlib>
#include <vector>
#include <algorithm>
using std::ofstream;
using std::string;
using std::vector;
using std::ios;
using std::cout;
using std::endl;
using std::ifstream;
using std::cin;
using std::regex;
using std::smatch;
using std::stoi;

int main()
{
```

```
23          ofstream fileA("A.dat",ios::out|ios::binary);
24          ofstream fileB("B.dat",ios::out|ios::binary);
25          if(!fileA.is_open()||!fileB.is_open()) assert(0);
26          string str_buf;
27          cout<<"Type the A's available workhours, and end with 'A'."<<endl;
28          getline(cin,str_buf,'A');
29          fileA.write(&str_buf[0],str_buf.length());
30          fileA.close();
31          str_buf.clear();
32          cin.ignore();
33          cout<<"Type the B's available workhours, and end with 'B'."<<endl;
34          getline(cin,str_buf,'B');
35          fileB.write(&str_buf[0],str_buf.length());
36          fileB.close();
37          ifstream in_fileA("A.dat",ios::in|ios::binary);
38          ifstream in_fileB("B.dat",ios::in|ios::binary);
39          if(!in_fileA.is_open()||!in_fileB.is_open()) assert(0);
40          str_buf.resize(200);
41          in_fileA.read(&str_buf[0],200);
42          regex reg("([0-9]{1,2}):00~([0-9]{1,2}):00");
43          smatch sm;
44          vector<int> a_times,b_times;
45          while(regex_search(str_buf,sm,reg))
46          {
47              for(int i=stoi(sm[1]);i<stoi(sm[2]);i++)
48                  a_times.push_back(i);
49              str_buf = sm.suffix().str();
50          }
51          str_buf.clear();
52          str_buf.resize(200);
53          in_fileB.read(&str_buf[0],200);
54          while(regex_search(str_buf,sm,reg))
55          {
56              for(int i=stoi(sm[1]);i<stoi(sm[2]);i++)
57                  b_times.push_back(i);
58              str_buf = sm.suffix().str();
59          }
60          vector<int> res(25);
61          vector<int>::iterator it=std::set_intersection(a_times.begin(),a_times.end(),
```

```
62          res.resize(it-res.begin());
63          bool first_flag=true;
64          int last_val=0;
65          std::ostringstream res_buf;
66          for(auto it=res.begin();it!=res.end();it++)
67          {
68              if(first_flag)
69              {
70                  first_flag=false;
71                  res_buf<<*it<<":00~";
72                  last_val=*it;
73              }
74              else if(last_val+1!=*it)
75              {
76                  res_buf<<last_val+1<<":00,";
77                  res_buf<<*it<<":00~";
78                  last_val=*it;
79              }
80              else
81              {
82                  last_val++;
83              }
84          }
85          res_buf<<last_val+1<<":00";
86          ofstream fileC("C.dat",ios::out|ios::binary);
87          fileC.write(&(res_buf.str())[0],res_buf.str().length());
88          fileC.close();
89      }
```

2.3 员工管理

```
1   #include <iostream>
2   #include <vector>
3   #include <string>
4   #include <cassert>
5   #include <fstream>
6   #include <regex>
7   #include <iomanip>
8   using std::string;
```

```
9     using std::vector;
10    using std::ios;
11    using std::cout;
12    using std::endl;
13    using std::cin;
14    using std::regex;
15    using std::smatch;
16    using std::setw;
17    #define BACKOFFSET (-(sizeof("ID:□□□□,Name:□□□□□□□□□,Phone:□□□□□□□□□□,Post□ID:□□□□
18    class Worker
19    {
20    public:
21        Worker(string ID,string name, string phone,string post,string address);
22        string& get_id();
23    private:
24        string _ID;
25        string _name;
26        string _phone;
27        string _post;
28        string _address;
29    friend std::ostream& operator <<(std::ostream& out, const Worker& x);
30    };
31    std::ostream& operator <<(std::ostream& out, const Worker& x)
32    {
33        out.setf(ios::left);
34        out<<"ID:"<<setw(4)<<x._ID;
35        out<<",Name:"<<setw(9)<<x._name;
36        out<<",Phone:"<<setw(9)<<x._phone;
37        out<<",Post□ID:"<<setw(8)<<x._post;
38        out<<",Address:"<<setw(14)<<x._address;
39        return out;
40    }
41
42    class AddressBook
43    {
44    public:
45        AddressBook();
46        ~AddressBook();
47        void add(Worker &worker);
```

```
48          Worker search(string id);
49          void modify(Worker &worker);
50
51      private:
52          std::fstream file;
53          regex reg;
54      };
55
56
57
58      int main()
59      {
60          Worker list[]=
61          {
62              {"1","Wang","150","123","Shanghai"},
63              {"2","Li","139","123","Shanghai"},
64              {"3","Zhao","110","321","Beijing"},
65              {"4","Qian","189","101","Nanjing"},
66              {"5","Sun","150","123","Shanghai"},
67          };
68          AddressBook book;
69          for(int i=0;i<5;i++)
70          {
71              book.add(list[i]);
72          }
73          cout<<book.search("1")<<endl;
74          Worker wgai("5","Liu","150","123","Shandong");
75          book.modify(wgai);
76          Worker wgai2("2","Kai","150","123","Tianjing");
77          book.modify(wgai2);
78          cout<<book.search("2")<<endl;
79          cout<<book.search("10")<<endl;
80      }
81
82      Worker::Worker(string ID,string name, string phone,string post,string address)
83          :_ID(ID),_name(name),_phone(phone),_post(post),_address(address){}
84
85      string& Worker::get_id()
86      {
```

```
87          return _ID;
88      }
89
90      AddressBook::AddressBook()
91          :reg("ID:([0-9]+)□{0,10},Name:([A-Za-z]+)□{0,10},Phone:([0-9]+)□{0,10},Post□I
92      {
93          file.open("workers.txt",ios::out|ios::in);
94          if(!file.is_open())
95          {
96                  file.open("workers.txt",ios::out);
97                  assert(file.is_open());
98                  file.close();
99                  file.open("workers.txt",ios::out|ios::in);
100                 assert(file.is_open());
101         }
102     }
103
104     AddressBook::~AddressBook()
105     {
106         assert(file.is_open());
107         file.close();
108     }
109
110     void AddressBook::add(Worker &worker)
111     {
112         file<<worker<<endl;
113     }
114
115     Worker AddressBook::search(string id)
116     {
117         string buf;
118         file.seekg(ios::beg);
119         getline(file,buf);
120         smatch sm;
121         while(!file.eof())
122         {
123             if(regex_search(buf,sm,reg))
124             {
125                 if(sm[1]==id)
```

```
126                 {
127                     return (Worker){sm[1],sm[2],sm[3],sm[4],sm[5]};
128                 }
129             }
130             buf.clear();
131             getline(file,buf);
132         }
133         file.seekg(ios::end);
134         cout<<"No this worker!"<<endl;
135         return (Worker){"","","","","",""};
136     }
137
138     void AddressBook::modify(Worker &worker)
139     {
140         string buf;
141         file.seekg(ios::beg);
142         getline(file,buf);
143         smatch sm;
144         while(!file.eof())
145         {
146             if(regex_search(buf,sm,reg))
147             {
148                 if(sm[1]==worker.get_id())
149                 {
150                     file.seekg(BACKOFFSET,ios::cur);
151                     file<<worker;
152                     file.seekg(ios::end);
153                     return;
154                 }
155             }
156             buf.clear();
157             getline(file,buf);
158         }
159         file.seekg(ios::end);
160         cout<<"No this worker!"<<endl;
161     }
```