ThoughtWorks 编程作业

README 文档用 Markdown 格式书写，用 Pandoc 生成了 PDF 版本。

代码和文档可以在 GitHub 上查看：district10/ThoughtWorksHomework20161013。

代码使用方法

使用 cmake 生成 Makefile 工程，然后 make：

```
$ cd ThoughtWorksHomework20161013
$ mkdir build && cd build
$ cmake ..
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tzx/git/ThoughtWorksHomework
20161013/build

$ make
Scanning dependencies of target demo
[ 25%] Building CXX object CMakeFiles/demo.dir/src/demo.cpp.o
Linking CXX executable demo
[ 25%] Built target demo
Scanning dependencies of target stdin2stdout
[ 50%] Building CXX object CMakeFiles/stdin2stdout.dir/src/stdin2stdout.
cpp.o
Linking CXX executable stdin2stdout
[ 50%] Built target stdin2stdout
Scanning dependencies of target test1
[ 75%] Building CXX object CMakeFiles/test1.dir/tests/test1.cpp.o
Linking CXX executable test1
[ 75%] Built target test1
Scanning dependencies of target test2
[100%] Building CXX object CMakeFiles/test2.dir/tests/test2.cpp.o
Linking CXX executable test2
[100%] Built target test2
```

Windows 上使用 CMake-GUI 生成 Visual Studio 工程，然后打开 sln 文件编译运行，其余类似，这里不在赘述。

例子程序

demo.cpp

demo.cpp（将会编译为 demo.exe）展示了题目中样本数据的输入和结果的输出。

```
$ ./demo
```

运行结果为：

```
input:

---
2016-06-02 20:00~22:00 7
2016-06-03 09:00~12:00 14
2016-06-04 14:00~17:00 22
2016-06-05 19:00~22:00 3
2016-06-06 12:00~15:00 15
2016-06-07 15:00~17:00 12
2016-06-08 10:00~13:00 19
2016-06-09 16:00~18:00 16
2016-06-10 20:00~22:00 5
2016-06-11 13:00~15:00 11
...

output:

---
[Summary]

2016-06-02 20:00~22:00 +210 -240 -30
2016-06-03 09:00~12:00 +420 -180 +240
2016-06-04 14:00~17:00 +660 -600 +60
2016-06-05 19:00~22:00 +0 -0 0
2016-06-06 12:00~15:00 +450 -300 +150
2016-06-07 15:00~17:00 +360 -200 +160
2016-06-08 10:00~13:00 +570 -330 +240
2016-06-09 16:00~18:00 +480 -300 +180
2016-06-10 20:00~22:00 +150 -120 +30
2016-06-11 13:00~15:00 +330 -200 +130

Total Income: 3630
Total Payment: 2470
Profit: 1160
...
```

可以看到，它正确地重现了题目中的样本输入和样本输出。

stdin2stdout.cpp

stdin2stdout.cpp（将会被编译为 stdin2stdout.exe）的输入为标准输入流，结果输出到标准输出流。

可以这样使用它：

```
$ cat ../../inputs/demo.txt | ./stdin2stdout.exe
```

```
[Summary]

2016-06-02 20:00~22:00 +210 -240 -30
2016-06-03 09:00~12:00 +420 -180 +240
2016-06-04 14:00~17:00 +660 -600 +60
2016-06-05 19:00~22:00 +0 -0 0
2016-06-06 12:00~15:00 +450 -300 +150
2016-06-07 15:00~17:00 +360 -200 +160
2016-06-08 10:00~13:00 +570 -330 +240
2016-06-09 16:00~18:00 +480 -300 +180
2016-06-10 20:00~22:00 +150 -120 +30
2016-06-11 13:00~15:00 +330 -200 +130

Total Income: 3630
Total Payment: 2470
Profit: 1160
```

这里的 ../../inputs/demo.txt 就是题目中的样本数据。

程序说明

generateSummary 函数在 GenerateSummary.h 中实现。 它的输入是一个字符串，其中每一行代表一个交易（Transaction）， 输出为题设要求的 Summary 信息。代码不多，我添加了必要的注释：

```
#ifndef GENERATE_SUMMARY_H
#define GENERATE_SUMMARY_H

#include <string>
#include <vector>
#include "Utils.h"
#include "Transaction.h"

// 打印交易信息，收支和利润
// 从 input 读入，输出到 output，都是 string 类型
std::string generateSummary(const std::string &input) {
    using namespace std;
    int income = 0, payment = 0, profit = 0;
    vector<string> transactions = unpackTransactions(input);
    string output = "[Summary]\n\n";
    for (int i = 0; i < transactions.size(); ++i) {
        Transaction t;
        t.parse(transactions[i]);
        income  += t.gain;
        payment += t.cost;
        profit  += t.net;
        output  += t.toString(); output += "\n";
    }
    output += "\n";
    char buf[100];
    snprintf( buf, sizeof(buf), "Total Income: %d\n",  income  ); output += buf;
    snprintf( buf, sizeof(buf), "Total Payment: %d\n", payment ); output += buf;
    snprintf( buf, sizeof(buf), "Profit: %d\n",        profit  ); output += buf;
    return output;
}

#endif // GENERATE_SUMMARY_H
```

上面，Transaciton 的 parse 函数，实现了对输入的读取，以及利润的计算。

- gain 是小明从报名羽毛球的人那里收取的费用
- cost 是球场的租金
- net 是小明的净利润

下面是 Transaction.h，是上面 parse 函数实现的源码：

```
#ifndef TRANSACTION_H
#define TRANSACTION_H

#include <ctime>
#include <cstdio>
#include <string>

// 价格表，9 点到 22 点，第一行为工作日，第二行为周末
static const int price_table[][24] = {
    // 9 ~ 12          12 ~ 18          18 ~ 22
    { 30,30,30, 50,50,50,50,50,50, 80,80,60,60 }, // mon~fri
    { 40,40,40, 50,50,50,50,50,50, 60,60,60,60 }, // sat,sun
};

// 交易信息
class Transaction {
public:
    int year, month, day;    // date
    int hour0, hour1;        // duration: hour0~hour1
    int npeople;             // #people
    int gain, cost, net;
    // 读取交易信息，格式为 "2016-06-02 20:00~22:00 7"
    // 并计算交易的收支和利润
    void parse(const std::string &input) {
        year    = str2int(input, 0, 4);
        month   = str2int(input, 5, 7);
        day     = str2int(input, 8, 10);
        hour0   = str2int(input, 11, 13);
        hour1   = str2int(input, 17, 19);
        npeople = str2int(input, 23, input.size());

        int ntable = calcNumOfTable(npeople);
        int tab = isWeekend(year, month, day) ? 1 : 0;
        gain = ntable == 0 ? 0 : npeople * 30;
        cost = 0;
        for (int i = hour0; i < hour1; ++i) {
            cost += ntable * price_table[tab][i - 9];
        }
        net = gain - cost;
    }

    // 打印收入、支出、利润
    std::string toString() {
        std::string ret;
        char buf[100];
        snprintf( buf, sizeof(buf),
                  "%04d-%02d-%02d %02d:00~%02d:00 +%d -%d",
                  year, month, day, hour0, hour1, gain, cost );
        ret += buf;
        if (net == 0) {
            snprintf( buf, sizeof(buf), " %d", net);
        } else if (net > 0) {
            snprintf( buf, sizeof(buf), " +%d", net);
        } else {
            snprintf( buf, sizeof(buf), " %d", net);
        }
        ret += buf;
        return ret;
    }

private:
    // 判断某年某月某日是否是周末
    bool isWeekend(int year, int month, int day) {
        std::tm date;
        date.tm_year = year - 1900;
        date.tm_mon  = month - 1;
        date.tm_mday = day;
        date.tm_hour = 12;
        std::mktime( &date );
        int dw = date.tm_wday;            // day of the week
        return dw == 0 || dw == 6;        // is sun, or sat?
    }

    // 根据报名人数计算需要借的羽毛球场的数量
    int calcNumOfTable(int m) {
        int ret = 0;
        int T = m / 6;
        int X = m % 6;
        switch (T) {
        case 0:
            ret = X < 4 ? 0 : 1;
            break;
        case 1:
            ret = 2;
            break;
        case 2: case 3:
            ret = X < 4 ? T : T + 1;
            break;
        default:
            ret = T;
        }
        return ret;
    }

    // 计算字符串 str[start, end] 对应的数字
    int str2int(const std::string &str, int start, int end) {
        int ret = 0;
        for (int i = start; i < end; ++i) {
            ret = ret * 10 + (str.at(i) - '0');
        }
        return ret;
    }
};

#endif // TRANSACTION_H
```

Utils.h 有几个工具性质的小函数:

```
#ifndef UTILS_H
#define UTILS_H

// 一些工具

#include <vector>
#include <string>
#include <iostream>
#include <fstream>
```

```
// 从流中获取交易信息，格式为 "2016-06-02 20:00~22:00 7"，并打包，分隔符为
"\n"
std::string getTransactionInfo(std::istream &is = std::cin) {
    using namespace std;
    string ret, line;
    while (getline(is, line)) {
        if (line.at(4) == '-' && line.at(16) == '~') { // 进行简单的校检
            ret += line + "\n";
        }
    }
    return ret;
}

// 把打包好的交易信息拆开成一条一条，用 vector 组织起来
std::vector<std::string> unpackTransactions(const std::string &info) {
    using namespace std;
    vector<string> ret;
    size_t last = 0, index;
    while ( (index = info.find_first_of('\n', last)) != string::npos )
{
        ret.push_back(info.substr(last, index-last));
        last = index + 1;
    }
    return ret;
}

#endif // UTILS_H
```