**匹配需求问题源码说明**

# 基本说明

代码采用C++编写，并采用了一些C++11的特性（主要是文件输入输出流部分），因此需要使用支持C++11的编译器编译，可使用Visual Studio 2017顺利编译并运行。

# 文件说明

Main.cpp ：主函数

Exchange\_Item.h ：（打款人/收款人）类定义和类成员函数声明

Exchange\_Item.cpp ：类成员函数的定义

requirematch.h ：匹配需求算法核心函数的声明

requirematch.cpp ：匹配需求算法核心函数的定义

CSVRow.h ：从CSV读入数据的类定义

SubSetSUM.h ：子集和问题算法的声明（匹配需求问题中没有用到）

SubSetSUM.cpp ：子集和问题算法的定义（匹配需求问题中没有用到）

## 2.1 Exchange\_Item.h

|  |
| --- |
| #pragma once  #ifndef EXCHANGEITEM  #define EXCHANGEITEM  #include<string>  #include<vector>  #include<map>  class Exchange\_Item  {  public:  Exchange\_Item() : User\_ID("000000"), money(0), initOrder(0), res\_money(0), weight(0), max\_weight(5) {};//构造函数  Exchange\_Item(const std::string str, int m, int order, int res\_m, int w, int max\_w) : User\_ID(str), money(m), initOrder(order), res\_money(res\_m), weight(w), max\_weight(max\_w) {};//构造函数  //构造函数  Exchange\_Item(const std::string str, int m) :User\_ID(str), money(m), initOrder(0), res\_money(m), weight(0), max\_weight(4) {};  //构造函数  Exchange\_Item(const std::string str, int m, int order) : User\_ID(str), money(m), initOrder(order), res\_money(m), weight(0), max\_weight(4) {};  //显示声明引用  Exchange\_Item(const Exchange\_Item&) = default;  Exchange\_Item& operator=(const Exchange\_Item&) = default;  bool setID(std::string str);//设置ID  std::string getID() const; //获取ID  bool setMoney(int m);//设置金额  int getMoney() const; //获取金额  int getInitOrder() const; //获取初始顺序值  bool setWeight(int w);//设置权重  int getWeight() const; //获取权重  bool weightIncrease();//权重+1  bool weightIncrease(int i); //权重+i  bool setResMoney(int res);//设置余额  int getResMoney() const; //获取余额  bool addExchangeInfo(std::string str, int m);//增加交易信息  std::map<std::string, int> getExchangeInfo();//获取交易信息  std::vector<std::string> getExchangeID();//获取交易信息中的ID  std::vector<int> getExchangeMoney();//获取交易信息中的金额  int maxWeight() const;//设置最大权重  void print();//打印对象信息  private:  std::string User\_ID;  int money;  int initOrder;//初始顺序  int res\_money;//余额  int weight;//权重  int max\_weight;//最大权重  std::vector<std::string> exchange\_ID; //交易信息中的ID  std::vector<int> exchange\_money; //交易信息中的金额  };  #endif // !EXCHANGEITEM ; |

## 2.2 requirematch.h

|  |
| --- |
| bool compID(const Exchange\_Item& Item1, const Exchange\_Item& Item2);  //比较用户ID，也就是字符串比较，第一个比第二个小则返回true  bool compOrder(const Exchange\_Item& Item1, const Exchange\_Item& Item2);  //比较用户原始输入顺序，第一个的order比第二个order小则返回true  bool compItemNoMaxWeight(const Exchange\_Item& Item1, const Exchange\_Item& Item2);  //不考虑权重，仅仅比较余额，第一个余额小，则返回true  bool compItem(const Exchange\_Item& Item1, const Exchange\_Item& Item2);  //考虑权重进行比较  bool readCSVdata(ExContainer& Rers, const std::string csvfilename);  // 从CSV读取数据到list容器中  bool readCSVdata(ExContainer& Rers, const std::string csvfilename, int startOrder);  // 从CSV读取数据到list容器中并给定初始顺序标号  size\_t dealRepetition(ExContainer& Rers, ExContainer& Ters);  // 先排序，Rers和Ters中可以抵消的数据发生抵消并记录，返回抵消的笔数  difftype finishedNum(ExContainer& Rers);  // 获取liast中余额为0个个数  bool has2sum(Itertype& first, Itertype& last, Exchange\_Item& target, Itertype& it1, Itertype& it2);  // 在迭代器指定范围[first,last)内，查找是否有两个元素的余额的和与target的余额相等  //如果有则将指向两元素迭代器赋给it1和it2，并返回true  void exchangeFunc(ExContainer& largeCont, ExContainer& smallCont,  ExSaver & largeSaver, ExSaver & smallSaver);  //两个容器中元素发生交易，将完成交易的存储在另一容器并从原先容器中删除  std::vector<int> getExTimes(ExSaver& Rers);  // 返回每个元素的交易次数的容器  int sumExTimes(ExSaver& Rers);  // 返回总交易次数  ExSaver::iterator maxExTimes(ExSaver& Rers);  // 返回最大交易次数  void writeResult(ExSaver& Rers, std::string result\_file);  //将结果写进CSV  void writeLog(std::string result\_log, ExSaver& Rers, ExSaver& Ters, double t);  //将运行时间，最大交易次数，总交易次数写进log |

## 2.3 SubSetSUM.h

|  |
| --- |
| void printSubsetSum(const std::vector<int> & w, const std::vector<bool>& x);  //输出已经找到的符合要求的子集  bool subsetSum(const std::vector<int>& w, std::vector<bool> x, int sum, int targetsum, size\_t k);  //寻找子集和的函数，先寻找元素较多的子集  bool subsetSum(const std::vector<int>& w, std::vector<bool> x, int targetsum, int n);  ////寻找子集和的函数，先寻找元素较少的子集  void test\_subsetsum();  //测试 printSubsetSum函数，subsetSum函数 |