项目说明文档

离散数学课程项目

——命题逻辑推理

作 者 姓 名： 陈奕名

学 号： 2351883

指 导 教 师： 李冰

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 项目分析

## 1.1 项目要求

根据下面命题，用命题逻辑推理方法确定谁是作案者，并给出推理过程，C语言源代码及演示界面。

（1）营业员A或B偷了手表；

（2）若A作案，则作案不在营业时间；

（3）若B提供的证据正确，则货柜未上锁；

（4）若B提供的证据不正确，则作案发生在营业时间；

（5）货柜上了锁。

# 2 项目设计

## 2.1 算法设计

由项目分析可以得出，该项目需要完成一次推理的模拟。在推理过程中，我们采用 A,B,C,D,E 分别代表五个命题:

A：营业员 A 偷了手表

B：营业员 B 偷了手表

C：作案不在营业时间

D：营业员 B 提供的证据正确

E：货柜未上锁

判别组合是否符合情况的表达式为：(A || B) && (!A || C) && (D || E) && (！D || !C) && !E 故可以把每一种情况转换为一个二进制序列，代表真假的一种组合，然后代入到表达式判断。每个命题均有两种可能，将每个命题的不同取值进行组合，若结果为真，则得出一种推理结果，将其输出。该过程利用五个循环来执行，让A、B、C、D、E 分别取到 0 和 1 进行判断，最后输出使上述表达式为真的A B 的取值

# 3 代码实现

#include <iostream>

using namespace std;

int main()

{

cout << "命题变元表示如下：" << endl;

cout << "A : 营业员A偷了手表" << endl;

cout << "B : 营业员B偷了手表" << endl;

cout << "C : 作案不在营业时间" << endl;

cout << "D : B提供的证据正确" << endl;

cout << "E : 货柜未上锁" << endl;

cout << endl;

cout << "命题与命题变元的对应表示为：" << endl;

cout << "（1）营业员A或B偷了手表 -> A | B " << endl;

cout << "（2）若A作案，则作案不在营业时间 -> A -> C" << endl;

cout << "（3）若B提供的证据正确，则货柜未上锁 -> D -> E" << endl;

cout << "（4）若B提供的证据不正确，则作案发生在营业时间 -> !D->!C" << endl;

cout << "（5）货柜上了锁 -> !E" << endl;

for (int A = 0; A <= 1; A++)

for (int B = 0; B <= 1; B++)

for (int C = 0; C <= 1; C++)

for (int D = 0; D <= 1; D++)

for (int E = 0; E <= 1; E++)

if ((A || B) && (!A || C) && (!D || E) && (D || !C) && !E)

{

cout << endl << "A=" << A << "，B=" << B << endl;

cout << endl;

if(A) cout << "A偷了手表";

if(B) cout << "B偷了手表";

cout << endl;

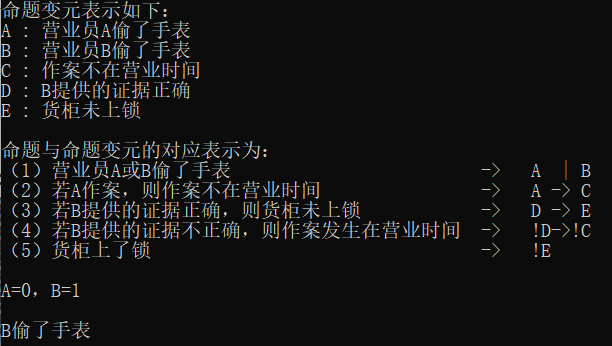
}

return 0;

}

# 4 测试

**实验结果:**



# 5 心得体会

这段程序通过布尔逻辑的方式，使用命题变元来表达复杂的逻辑推理问题，并通过枚举所有可能的命题值来寻找符合所有条件的解。程序的设计思路简单且有效，展示了如何利用命题逻辑解决实际问题中的推理和判断。

通过这个程序，我深入理解了命题逻辑与编程实现之间的关系。命题变元（A、B、C、D、E）代表了实际问题中的各种假设或条件，而布尔运算（与、或、非）则通过编程语言中的逻辑运算符来实现这些条件的组合。通过遍历所有可能的命题取值组合，并使用逻辑条件进行筛选，我们能够得出满足特定条件的情况。

此外，我还体会到了程序设计中枚举所有可能解的技巧。在本程序中，利用五重循环来模拟所有可能的命题取值组合，然后通过逻辑表达式进行过滤，这种穷举法的思路在解决类似的逻辑推理问题时非常有效。然而，随着问题规模的增大，枚举方法的效率可能会下降，这也提醒我在面对更复杂的问题时，可能需要更高效的算法来优化性能。

总的来说，这段程序帮助我理解了命题逻辑的基本应用，并让我意识到在程序设计中如何通过逻辑运算来模拟现实世界中的推理过程。