

**实 验 报 告**

**（2019 / 2020 学年 第 一 学期）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 离散数学 | | | | | |
| 实验名称 | 利用真值表法求取主析取范式以及主合取范式的实现 | | | | | |
| 实验时间 | 2020 | 年 | 10 | 月 | 28 | 日 |
| 指导单位 | 计算机学院计算机科学与技术系 | | | | | |
| 指导教师 | 柯昌博 | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 | 茹兴奥 | 班级学号 | B19030824 |
| 学院(系) | 计算机学院 | 专 业 | 信息安全 |

**实 验 报 告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称** | 利用真值表法求取主析取范式以及主合取范式的实现 | | | **指导教师** | 柯昌博 |
| **实验类型** | **验证** | **实验学时** | **4** | **实验时间** | 10.28 |
| 1. **实验目的和要求**   输入可以是命题变元，也可以是命题公式，输出是主吸取范式和主合取范式，同时输出真值表，方便验证。编程语言不限。 | | | | | |
| 二、**实验环境(实验设备)**  硬件：微型计算机  软件：Windows 操作系统、Microsoft Visual C++6.0、Java等可视化编程语言 | | | | | |
| **三、实验原理及内容**   |  | | --- | | #include<cmath> | |  | #include <iostream> | |  | #include <set> | |  | #include <stack> | |  | #include <string> | |  | #include <vector> | |  | using std::set; | |  | using std::vector; | |  | using std::string; | |  | using std::cin; | |  | using std::cout; | |  | using std::endl; | |  |  | |  | bool cToB(char input) | |  | { | |  | return input == '1'; | |  | } | |  | char bToC(bool input) | |  | { | |  | return input ? 1 : 0; | |  | } | |  | string bToS(bool input) | |  | { | |  | return input ? "1" : "0"; | |  | } | |  |  | |  | int strReplace(string& str, const string& src, const string& dest) | |  | { | |  | int counter = 0; | |  | string::size\_type pos = 0; | |  | while ((pos = str.find(src, pos)) != string::npos) { | |  | str.replace(pos, src.size(), dest); | |  | ++counter; | |  | pos += dest.size(); | |  | } | |  | return counter; | |  | } | |  |  | |  | string inffixToSuffix(string input) | |  | { | |  | std::stack<char> s; | |  | string output = ""; | |  | for (string::iterator c = input.begin(); c != input.end(); ++c) { | |  | switch (\*c) { | |  | case '1': | |  | case '0': | |  | output += \*c; | |  | break; | |  | case '(': | |  | s.push(\*c); | |  | break; | |  | case ')': | |  | while (!s.empty()) { | |  | if (s.top() == '(') { | |  | s.pop(); | |  | break; | |  | } | |  | output += s.top(); | |  | s.pop(); | |  | } | |  | break; | |  | case '!': | |  | s.push(\*c); | |  | break; | |  | case '|': | |  | case '&': | |  | while (!s.empty() && s.top() != '(' && s.top() != '>' && s.top() != '=') { | |  | output += s.top(); | |  | s.pop(); | |  | } | |  | s.push(\*c); | |  | break; | |  | case '>': | |  | case '=': | |  | while (!s.empty() && s.top() != '(') { | |  | output += s.top(); | |  | s.pop(); | |  | } | |  | s.push(\*c); | |  | break; | |  | } | |  | } | |  | while (!s.empty()) { | |  | output += s.top(); | |  | s.pop(); | |  | } | |  | return output; | |  | } | |  |  | |  | bool calcSuffixExpr(string input) | |  | { | |  | std::stack<bool> s; | |  | for (string::iterator c = input.begin(); c != input.end(); ++c) { | |  | if (\*c == '0' || \*c == '1') { | |  | s.push(cToB(\*c)); | |  | } else if (\*c == '!') { | |  | bool x = s.top(); | |  | s.pop(); | |  | s.push(!x); | |  | } else { | |  | bool x = s.top(); | |  | s.pop(); | |  | bool y = s.top(); | |  | s.pop(); | |  | bool result; | |  | switch (\*c) { | |  | case '|': | |  | result = x | y; | |  | break; | |  | case '&': | |  | result = x & y; | |  | break; | |  | case '>': | |  | result = (!x) | y; | |  | break; | |  | case '=': | |  | result = (x == y); | |  | break; | |  | } | |  | s.push(result); | |  | } | |  | } | |  | return s.top(); | |  | } | |  |  | |  | void parseInput(string input, vector<string>& output) | |  | { | |  | string now; | |  | string symbols = "!()&|=>"; | |  | set<string> outputSet; | |  | for (string::iterator c = input.begin(); c != input.end(); ++c) { | |  | if (symbols.find(\*c) == string::npos) { | |  | now += \*c; | |  | } else { | |  | if (now.length() != 0) { | |  | outputSet.insert(now); | |  | now = ""; | |  | } | |  | } | |  | } | |  | if (now.length() != 0) { | |  | outputSet.insert(now); | |  | now = ""; | |  | } | |  | for (set<string>::iterator str = outputSet.begin(); str != outputSet.end(); ++str) { | |  | output.push\_back(\*str); | |  | } | |  | } | |  |  | |  | int main() | |  | { | |  | cout << "请使用 ! | & > =" << endl; | |  | cout << "hello,using ! | & > = please" << endl; | |  | string input; | |  | cin >> input; | |  | vector<string> varName; | |  | parseInput(input, varName); | |  | int varNum = varName.size(); | |  | int tableNum = (int)pow(2, varNum); | |  | bool table[tableNum][varNum + 1]; | |  |  | |  | for (int i = 0; i < tableNum; i++) { | |  | int num = i; | |  | for (int j = 0; j < varNum; j++) { | |  | table[i][j] = num % 2; | |  | num /= 2; | |  | } | |  | } | |  | cout << "真值表" << endl; | |  | cout << "Truth table" << endl; | |  | for (int i = 0; i < varNum; i++) { | |  | cout << varName[i] << '\t'; | |  | } | |  | cout << input << endl; | |  | for (int i = 0; i < varNum + 1; i++) { | |  | cout << "-\t"; | |  | } | |  | cout << endl; | |  | for (int i = 0; i < tableNum; i++) { | |  | string expr = input; | |  | for (int j = 0; j < varNum; j++) { | |  | strReplace(expr, varName[j], bToS(table[i][j])); | |  | cout << bToS(table[i][j]) << "\t"; | |  | } | |  | string suffixExpr = inffixToSuffix(expr); | |  | table[i][varNum] = calcSuffixExpr(suffixExpr); | |  | cout << table[i][varNum] << endl; | |  | } | |  |  | |  | cout << "主合取范式" << endl; | |  | cout << "Principal Conjunctive Normal Form" << endl; | |  | bool first = true; | |  | for (int i = 0; i < tableNum; i++) { | |  | if (table[i][varNum]) { | |  | continue; | |  | } | |  | if (first) { | |  | first = false; | |  | } else { | |  | cout << "&"; | |  | } | |  | cout << "("; | |  | for (int j = 0; j < varNum; j++) { | |  | if (table[i][j]) { | |  | cout << "!"; | |  | } | |  | cout << varName[j]; | |  | if (j != varNum - 1) { | |  | cout << "|"; | |  | } | |  | } | |  | cout << ")"; | |  | } | |  | cout << endl; | |  |  | |  | cout << "主析取范式" << endl; | |  | cout << "Principal Disjunctive Normal Form" << endl; | |  | first = true; | |  | for (int i = 0; i < tableNum; i++) { | |  | if (table[i][varNum] == false) { | |  | continue; | |  | } | |  | if (first) { | |  | first = false; | |  | } else { | |  | cout << "|"; | |  | } | |  | cout << "("; | |  | for (int j = 0; j < varNum; j++) { | |  | if (table[i][j] == false) { | |  | cout << "!"; | |  | } | |  | cout << varName[j]; | |  | if (j != varNum - 1) { | |  | cout << "&"; | |  | } | |  | } | |  | cout << ")"; | |  | } | |  | cout << endl; | |  |  | |  | return 0; | |  | } | | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **四、实验小结**（包括问题和解决方法、心得体会、意见与建议等）  中文五号宋体，英文五号Times new roman字体，1.25倍行距  **说明：这部分内容主要包括：在编程、调试或测试过程中遇到的问题及解决方法、本次实验的心得体会、进一步改进的设想等。**  (一)实验中遇到的主要问题及解决方法  如：在程序运行时，总出现报错的问题，后来用百度寻找解决方法，发现是中文符号所导致的错误，进行切换英文的修改后错误消除，从而说明敲代码要认真。  （二）实验心得  下次敲代码应该用心一点，避免犯一些低级错误。  （三）意见与建议（没有可省略） | | | | | |
| 1. **支撑毕业要求指标点**   支撑毕业要求的指标点为：   * 1-4掌握计算机科学与技术领域的专业知识，能将专业知识用于分析和解决计算机领域复杂工程问题。   √   * 2-1能够应用数学、自然科学和工程科学的基本知识，识别和分析计算机领域复杂工程问题的特征。 | | | | | |
| **六、指导教师评语 (含学生能力达成度的评价)** | | | | | |
| **成 绩** |  | **批阅人** |  | **日 期** |  |

如果不太想写太多字，“指导教师评语”也可以设计为如下的各选择项用打勾形式（仅仅作为一个简单示例，请各课程负责人根据课程和实验情况以及支撑的指标点来自行设定选择项，同一门课程的不同实验评分细则项允许存在不同）：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **评 分 细 则** | **评分项** | **优秀** | **良好** | **中等** | **合格** | **不合格** |
| **遵守实验室规章制度** |  |  |  |  |  |
| **学习态度** |  |  |  |  |  |
| **算法思想准备情况** |  |  |  |  |  |
| **程序设计能力** |  |  |  |  |  |
| **解决问题能力** |  |  |  |  |  |
| **课题功能实现情况** |  |  |  |  |  |
| **算法设计合理性** |  |  |  |  |  |
| **算法效能评价** |  |  |  |  |  |
| **回答问题准确度** |  |  |  |  |  |
| **报告书写认真程度** |  |  |  |  |  |
| **内容详实程度** |  |  |  |  |  |
| **文字表达熟练程度** |  |  |  |  |  |
| **其它评价意见** |  | | | | |
| **本次实验能力达成评价（总成绩）** |  |  |  |  |  |