



**《程序设计原理》实验报告**

**——卡牌大师**

**学 号 3022244290**

**姓 名 陈秋澄**

**学 院 智能与计算学部**

**年 级 22级**

**任课教师 李雪威**

**2022年12月10日**

# 实验目的

解决“二十四点”卡牌问题，输出所有符合条件的结果。

# 二．实验内容

1. **设计思路**

常见的解法有两种：一种是使用递归实现的主符号法；另一种是使用栈进行模拟，转成逆波兰表达式进行计算。两种方法都非常的经典，效率较高。我采用了代码较为简洁的以递归为主的解法。

关于两种方法的比较：我认为递归法对数学逻辑方面要求更高，代码长度较短；

而栈进行模拟的方法代码较长，但更易理解。共同点是两个方法效率都比较高。

1. **代码实现**

#include <cstdio>

#include <iostream>

#include <cstdlib>

#include <cstring>

#include <map>

#include <vector>

int n, a[7], vis[7];

int sti[7], stc[7], topi,topc;

std::vector<int> ex;

std::map<std::vector<int>, int> is; //使用map，便于查找

inline void calc() {

int b = sti[topi -- ], a = sti[topi -- ], opt = stc[topc -- ], res;

if (opt == 14) res = a + b;

if (opt == 15) res = a - b;

if (opt == 16) res = a \* b;

if (opt == 17) res = a / b;

sti[ ++ topi] = res;

}

//使用枚举法

inline bool check() {

if (is[ex]) return false;

is[ex] = true;

topi = 0, topc = 0;

for (int c: ex)

if (c <= 13) {

sti[ ++ topi] = c;

}

else {

while (topc && stc[topc] / 2 >= c / 2) calc();

stc[ ++ topc] = c;

}

while (topc) calc();

return sti[1] == 24;

}

inline void print() {

for (int c: ex)

if (c <= 13) printf("%d", c);

else {

if (c == 14) putchar('+');

if (c == 15) putchar('-');

if (c == 16) putchar('\*');

if (c == 17) putchar('/');

}

putchar('\n');

} //加减乘除转化为数值，以便求解

int dfs(int u) { //使用递归

if (u == n \* 2) return check() && (print(), 1);

int res = 0;

if (u & 1) {

for (int i = 1; i <= n; ++ i )

if (!vis[i]) {

vis[i] = true, ex.emplace\_back(a[i]);

res += dfs(u + 1);

vis[i] = false, ex.pop\_back();

}

}

else {

int i=14;

while ( i <= 17) {

ex.emplace\_back(i);

res += dfs(u + 1);

ex.pop\_back();

i++;

}

}

return res;

}

int priority(char ope\_){

if(ope\_=='+'||ope\_=='-')

return 1;

else if (ope\_=='\*'||ope\_=='/')

return 2;

else

return 0;

}

int main() {

scanf("%d", &n);

int i = 1;

while(i <= n) {

scanf("%d", a + i);

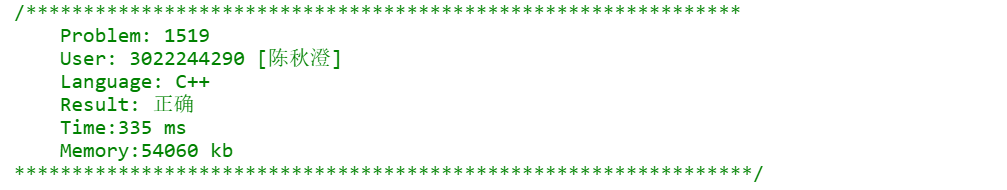
++ i ;

}

printf("%d\n", dfs(1)); //输出结果

}

1. **结果展示**

****

# 三．实验中遇到的问题和解决办法

问题：方法选择不当。我写的方法二代码长度有近二百行，不便于阅读和debug。

解决办法：复习递归相关知识，放弃方法二（即栈进行模拟，转成逆波兰表达式进行计算）。

关于两种方法的比较：我认为递归法对数学逻辑方面要求更高，代码长度较短；

而栈进行模拟的方法代码较长，但更易理解。共同点是两个方法效率都比较高。