# 实验 函数

樊子谦

软件学院 软件工程 2024级1班

1. 编写一个函数multiple，用于确定一对整数中第二个整数是否是第一个整数的倍数。函数应该需要两个整数参数，并且如果第二个整数是第一个整数的倍数的话，返回1，否则返回0。在程序中使用这个函数判断输入的一系列整数对

假如第二个整数b是第一个数a的倍数，那么b%a结果一定为0.直接返回not (b%a)即可满足题意。

题面要求输入一系列整数，那么选择使用while处理。

#include <bits/stdc++.h>

using namespace std;

bool multiple(int a, int b) {

return !(b % a);

}

int main() {

int a, b;

while (cin >> a >> b) {

bool is\_multiple = multiple(a, b);

if (is\_multiple) {

cout << b << " is a multiple of " << a << endl;

} else {

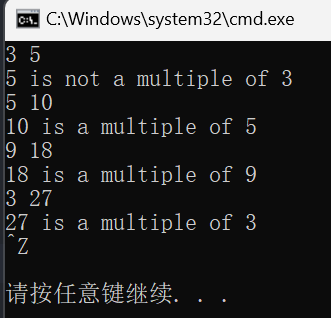
cout << b << " is not a multiple of " << a << endl;

}

}

return 0;

}



【2】                      编写一个程序求一系列整数之和。假定输入的第一个整数制定了继续输入的值的个数，比如输入 5 100 200 300 400 500，代表计算5个数的和，总和是1500。

就是读入个数然后输入n个数，边输入边累加即可。

但考虑到实验题目为“函数”，这样做好像和函数没什么关系，于是编写一个函数sum用来求和。传入参数是一个vector。Vector可以用迭代器遍历，所以函数没有必要传入n。

#include <bits/stdc++.h>

using namespace std;

int sum(const auto &a) {

int ret = 0;

for (const auto &i : a) {

ret += i;

}

return ret;

}

int main() {

int n;

cin >> n;

vector<int> a(n);

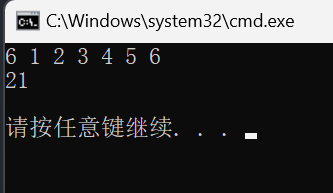
for (int i = 0; i < n; i++)

cin >> a[i];

cout << sum(a) << endl;

return 0;

}



【3】                      现有21根火柴，两人轮流取，每人每次可以取走1至4根，不可多取，也不能不取，谁取最后一楰火柴谁输。任务一：请编写一个程序进行人机对弈，要求人先取，计算机后取（随机取）；任务二：计算机一方为“常胜将军”，即找出必胜策略。

一道很经典的博弈论问题。一眼能看出是反nim游戏。

参考：oiwiki-博弈论（<https://oi-wiki.org/math/game-theory/intro/>）

但是这里并不需要什么很复杂的理论知识。简单思考可以发现，这个游戏的过程要求中间步骤中，计算机按照一定的规律来控制火柴的性质，才能达到最优策略。在博弈论中，达成这个目的可以关注奇偶性、质数、倍数等性质。这里容易发现，后手方（计算机）可以控制每轮取火柴双方的根数和均为5，而21 mod 5 =1，那么这样计算机是必定赢的。

这里对游戏过程的模拟用了同一个函数（因为任务一和任务二有高度相似性），在计算机取的时候引用不同函数做特殊处理。

此外，引用了mt19937生成随机数。随机数 的生成在get\_rand\_num函数内实现。

在主函数内，如果不选择退出游戏，会反复调用main函数自身来达到“重新开始游戏”的效果。

#include <bits/stdc++.h>

using namespace std;

int get\_rand\_num(int lower, int upper) {

mt19937 rd(random\_device{}());

uniform\_int\_distribution<int> dist(lower, upper);

return dist(rd);

}

int stupid\_take(int num) {

return get\_rand\_num(1, min(4, num));

}

int clever\_take(int take) {

return 5 - take;

}

void game(int mode) {

int num = 21;

int round = 0;

while (num) {

round++;

cout << "Round " << round << ":\n";

cout << "Number of sticks left: " << num << '\n';

int take;

cout << "How many sticks do you want to take ? (1-" << min(4, num) << "): ";

while (cin >> take) {

if (take >= 1 && take <= min(4, num)) {

break;

} else {

cout << "Invalid input, please enter again: ";

}

}

num -= take;

if (num == 0) {

cout << "You lose. TWT " << endl;

break;

}

int computer\_take;

if (mode == 1)

computer\_take = stupid\_take(num);

else

computer\_take = clever\_take(take);

cout << "Computer takes " << computer\_take << " sticks." << endl;

num -= computer\_take;

if (num == 0) {

cout << "You win! ^\_^" << endl;

break;

}

}

}

int main() {

int mode;

cout << "Choose the mode:\n";

cout << "1. Stupid Computer\n2. Smart Computer\n";

cin >> mode;

game(mode);

bool new\_game;

cout << "New game? (1/0): ";

cin >> new\_game;

if (new\_game) {

system("cls");

main();

} else {

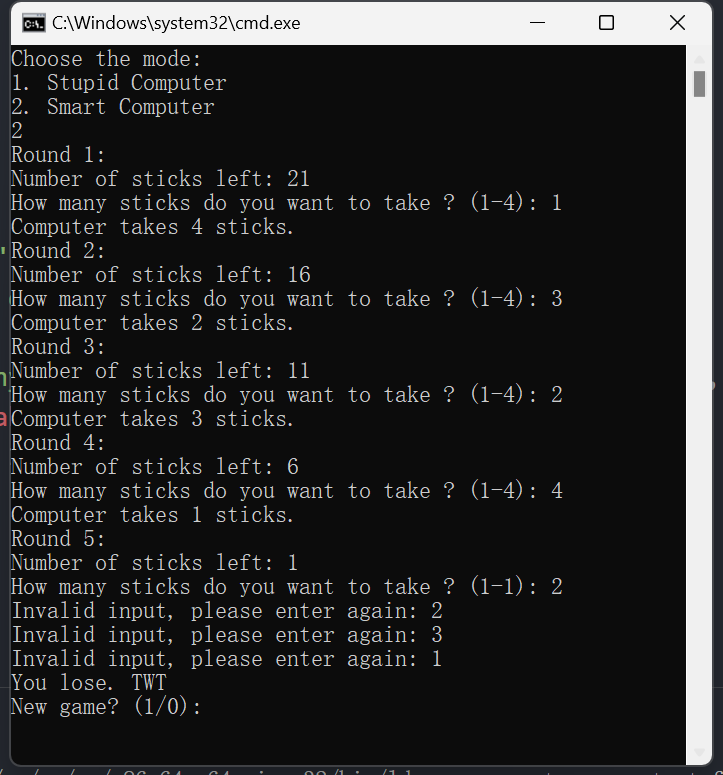
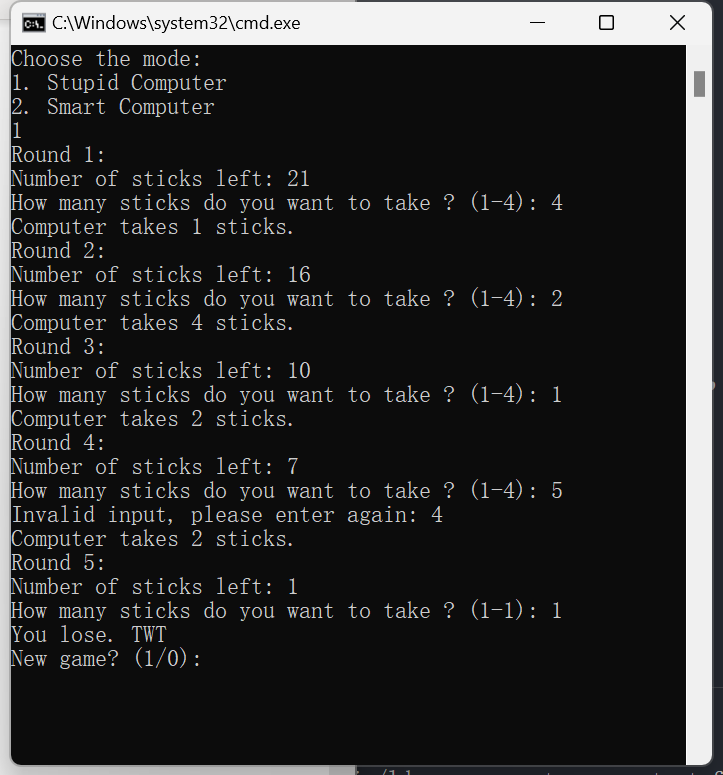
cout << "Goodbye!" << endl;

return 0;

}

return 0;

}

经过测试，程序具有良好的稳定性，每一轮会提示剩余的火柴，对用户错误的输出会提示非合法输出。

心得：

在完成这三个实验的过程中，我深刻体会到了编程中函数的重要性和实用性。实验一让我掌握了如何通过模运算来判断一个数是否是另一个数的倍数，并且通过封装逻辑到函数中，我学会了如何使代码更加简洁和易于理解。实验二则让我熟悉了使用`vector`和迭代器来编写通用函数，这不仅提高了代码的复用性，也让我对C++的容器和迭代器有了更深的认识。而实验三则是最具挑战性的，它不仅让我学习了如何编写游戏程序，还初步接触了博弈论中的策略，比如如何通过控制游戏状态来确保胜利。这个实验锻炼了我的逻辑思维和问题解决能力，并且让我意识到算法和策略在解决问题中的重要性。总的来说，这些实验不仅提升了我的编程技能，还让我对函数、容器、迭代器以及博弈论的应用有了更深的理解，我相信这些知识和技能将对我的未来学习和工作产生积极影响。