天津大学

程序设计综合实践课程报告

基础算法实验

学生姓名 陈昊昆

学院名称 智算学部

专　　业 软件工程

学　　号 **3021001196**

1. 士兵队列训练问题

1.1 题目分析

通过遍历数组，达到 2 号出列和 3 号出列的目标

设置一个计数器，初始值为 0

每访问一个未出列的位置（即非零位置），计数器+1

当计数器达到 1 时，出列（将当前位置置 0），即为 2 号出列

当计数器达到 2 时，出列（将当前位置置 0），即为 3 号出列

循环，直到非 0 位置总数<=3

1.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;  //共 n 行
    for (int i = 0; i < n; i++) {
        int num;
        cin >> num;  //每一行的人数
        int *array = new int[num];
        for(int j = 0; j < num; j++) array[j] = 1;
        int cnt = 0;
        int count = 0;
        //当总人数<=3 时结束循环
        while(num > 3){
            //2 号出列
            for (int j = 0; j < num; j++) {
                if (array[j] == 1) {
                    if (cnt == 0) {cnt++; count++;}
                    else {
                        cnt = 0;
                        array[j] = 0;
                    }
```

```
                }
            }
            if (count <= 3) break; //判断队中人数是否<=3
            count = 0;
            cnt = 0;

            //3 号出列
            for (int j = 0; j < num; j++) {
                if (array[j] == 1) {
                    if (cnt <= 1) {cnt++; count++;}
                    else {
                        cnt = 0;
                        array[j] = 0;
                    }
                }
            }
            if (count <= 3) break; //判断队中人数是否<=3
            count = 0;
            cnt = 0;
        }
        for (int j = 0; j < num; j++){
            if (array[j] == 1) cout << j+1 << " ";

        }
        cout << endl;
        delete array;
    }
}
```

2. 斐波那契数列

2.1 题目分析

设置一个类型为 long long 的数组，存放计算出来的数列

通过 for 循环，计算出每一项的值，使得该项的值为前两项的和

2.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;
long long f[60]; //设置全部数组遍历 long long 类型
void fobi();
int main(){
    fobi();
    int n;
    cin >> n;  //输入 n 个样例
    for (int i = 0; i < n; i++) {
        int m;
        cin >> m; //获取第 m-1 项的值
        cout << f[m-1] << endl;
    }
}

//计算出每一项的值，使得该项的值为前两项的和
void fobi(){
    f[0] = 1;
    f[1] = 1;
    for (int i = 2; i < 60; i++) f[i] = f[i-1] + f[i-2];
}
```

3. [数值问题]高精度加法

3.1 题目分析

设置三个 char 型数组，前两个 a,b 存放输入，后一个 c 用于输出

将数组 a,b 逆序，从而使末尾对齐

每位相加后，减去'0'的 ASC 码，即为所得

当出现进位时，使用 flag 变量，使下一位+1

为防止出现字符与空相加，先把较短数字的空项赋值为'0'

3.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;
int swap(char *p);

int main(){
    char a[200], b[200], c[201]; //存放输入输出的数组
    cin >> a >> b;
    //将数组 a,b 逆序
    int a1 = swap(a);
    int b1 = swap(b);
    int flag = 0;
    //为防止出现字符与空相加，先把较短数字的空项赋值为’0’
    if (a1 > b1) {
        for (int i = b1; i < a1; i++) b[i] = '0';
    }
    if (a1 < b1) {
        for (int i = a1; i < b1; i++) a[i] = '0';
    }
    //每位相加并设置进位
    for(int i = 0; i < 201; i++){
        if (a[i] == 0) a[i] = '0';
        if (b[i] == 0) b[i] = '0';
        if(a[i] == '0' && b[i] == '0' && flag == 0) a[i] = b[i] = 0;
        c[i] = a[i] + b[i] - '0';
        if(flag == 1) c[i] += 1;
```

```
        flag = 0;
        if(c[i] > '9'+10 || c[i] < '0' ) c[i] = 0;
        if(c[i] > '9' && c[i] <= '9' + 10) {
            c[i] = c[i] - 10;
            flag = 1;
        }
    }
    swap(c);
    cout << c;

}

//将数组逆序
int swap(char *p){
    int n = 0;
    for (int i = 0; i < 200; i++) if (p[i] == '\0'){
        n = i ;
        break;
    }
    for (int i = 0; i < n/2; i++){
        char tem = p[n - 1 - i];
        p[n - 1 - i] = p[i];
        p[i] = tem;
    }
    return n;
}
```

4. 三角形个数

4.1 题目分析

做三层 for 循环，即固定两个数，对第三个数循环

当满足两边之和大于第三边，两边之差小于第三边，就是三角形，计数+1

4.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;


int main(){
    int n;
    cin >> n;
    int* p = new int[n];
    for (int i = 0; i < n; i++) cin >> p[i];
    int cnt = 0;

    //做三层 for 循环，即固定两个数，对第三个数循环
    //当满足两边之和大于第三边，两边之差小于第三边，就是三角形，计数+1
    for (int i = 0; i < n-2; i++){
        for(int j = i+1; j < n-1; j++){
            for (int k = j+1; k < n; k++){
                if (p[i] + p[j] > p[k] && p[k] -p[i] < p[j]) cnt ++;
                }
            }
        }
    cout << cnt;
    delete p;
}
```

5. 找零

5.1 题目分析

符合最优子结构和贪心选择

利用贪心算法，先找面额大的纸币，来减少找零个数

对每种面额的纸币，先判断是否有该面额纸币，若有，则找出

5.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    for (int i = 0; i < n; i++){
        int t;
        cin >> t;
        int c[7];
        int cnt = 0;
        for (int j = 0; j < 7; j++) cin >> c[j];  //输入个面额纸币的数量

        //面额从大到小找出
        //先判断是否存在该面额的纸币
        while(t >= 100){
            if (c[6] > 0) {
                c[6]--;
                t = t -100;
                cnt++;
            }
            else break;
        }
        while(t >= 50){
            if (c[5] > 0) {
                c[5]--;
                t = t -50;
                cnt++;
```

```
            }
            else break;
        }
        while(t >= 20){
            if (c[4] > 0) {
                c[4]--;
                t = t -20;
                cnt++;
            }
            else break;
        }
        while(t >= 10){
            if (c[3] > 0) {
                c[3]--;
                t = t -10;
                cnt++;
            }
            else break;
        }
        while(t >= 5){
            if (c[2] > 0) {
                c[2]--;
                t = t -5;
                cnt++;
            }
            else break;
        }
        while(t >= 2){
            if (c[1] > 0) {
                c[1]--;
                t = t -2;
                cnt++;
            }
            else break;
        }
        while(t >= 1){
            if (c[0] > 0) {
                c[0]--;
                t = t -1;
                cnt++;
            }
            else break;
        }
        if (t == 0) cout << cnt << endl;
```

```
        else cout << -1 << endl;
    }
    }
```

6. 汉诺塔

6.1 题目分析

利用递归算法，计算次数

边界条件是直到只有一层时，搬运一次

否则搬运两次 n-1 层的塔 ＋1 次

写成尾递归，以转换成高效的循环

6.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

long long han(int n);

int main(){
    int n;
    cin >> n;
    for (int i = 0; i < n; i++){
        int j;
        cin >> j;
        cout << han(j) << endl;
    }

}

long long han(int n){
    if (n == 1) return 1;  //直到只有一层时，搬运一次
    else return 2*han(n - 1)+1; //搬运两次 n-1 层的塔 + 1 次
}
```

7. 绝对值排序

7.1 题目分析

写一个简单插入排序

将判断条件改为绝对值之间的判断

7.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    while (cin >> n){
        int *p = new int[n];
        for (int i = 0; i < n; i++){
            cin >> p[i];
        }

        for (int i = 1; i < n; i++){
            int j = i;
            int tmp = p[i];
            // 绝对值之间的判断
            while((abs(tmp) > abs(p[j-1])) && j > 0){
                p[j] = p[j-1];
                j--;
            }
            p[j] = tmp;
        }

        for (int i = 0; i < n; i++){
            cout << p[i] << " ";
        }

        delete []p;
    }
}
```