每周训练 2 萌新场题解

本次萌新场主要针对 c 语言的基础语法，包括输入输出、数据类型、条件分支、循环、数组等应用，也包括一些基础的算法思想，排序、二分等。萌新场目的是训练对编程语言的使用熟练程度，并不包含过多算法思想，因此这次题解不会出现过于详细的解释，有问题可以在群里一起探讨，

A

基础输入输出、数据类型。

```
#include<stdio.h>

int main() {
    double f;
    // 此处如果使用 float 可能会因精度 wa
    scanf("%lf", &f);
    printf("%.lf", 5 * (f - 32) / 9);
    return 0;
}
```

B

数组简单运用。

```
#include <stdio.h>

#define N 31
int a[N], b[N];

int main() {
    a[0] = 2;
    b[0] = 1;
    int n;
    scanf("%d", &n);
    double sum = 0;
    for (int i = 0; i < n; i++) {
        sum += (double) a[i] / (double) b[i];
        a[i + 1] = a[i] + b[i];
        b[i + 1] = a[i];
    }
    printf("%.4lf\n", sum);
}
```

C

还是数组的运用，循环模拟开关灯。

```
#include<stdio.h>

#define N 5005
```

```
int light[N] = {0};

int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    for (int i = 2; i <= m; i++) {
        for (int j = i; j <= n; j += i) {
            light[j] = !light[j];
        }
    }
    int first = 1;
    for (int i = 1; i <= n; i++) {
        if (!light[i]) {
            if (first)
                printf("%d", i), first = 0;
            else
                printf(",%d", i);
        }
    }
    return 0;
}
```

D

简单的字符串处理。

```
#include<stdio.h>

int main() {
    char c;
    while (scanf("%c", &c) != EOF) {
        if (c >= 'A' && c <= 'Z')
            printf("%c", c <= 'E' ? c + 21 : c - 5);
        else printf("%c", c);
    }
    return 0;
}
```

E

循环遍历一遍，统计出每个字母出现的次数，再循环第二遍，将第一个出现次数为 1 的字母输出即可。

```
#include<stdio.h>

#define N 100005
int cnt[N] = {0};
char str[N];

int main() {
    scanf("%s", str);
    for (int i = 0; str[i] != '\0'; i++)
        cnt[str[i] - 'a']++;
    for (int i = 0; str[i] != '\0'; i++) {
        if (cnt[str[i] - 'a'] == 1) {
            printf("%c", str[i]);
```

```
            return 0;
        }
    }
    printf("no");
    return 0;
}
```

F

字符串处理。

```
#include<stdio.h>

int main() {
    char c;
    int blank = 0;
    while (scanf("%c", &c) != EOF) {
        if (c != ' ')printf("%c", c), blank = 0;
        else {
            if (!blank)printf(" ");
            blank = 1;
        }
    }
    return 0;
}
```

G

二维数组的应用，根据题意，草只存在三种情况，横着相连，竖着相连和单独一棵，此时只需要分别计算横着和竖着相连的草，剔除相连的草后再遍历一遍得出单独一棵草的数目即可。（也可以使用 深度优先搜索DFS 来做）

```
#include<stdio.h>

#define N 105
char g[N][N];

int main() {
    int n, m, sum = 0;
    scanf("%d %d", &n, &m);
    for (int i = 1; i <= n; i++)
        scanf("%s", g[i] + 1);
    for (int i = 2; i <= n; i++) {
        // 先数竖着相连的草
        for (int j = 1; j <= m; j++) {
            if (g[i][j] == '#' && g[i - 1][j] == '#') {
                sum++;
                g[i][j] = g[i - 1][j] = '.';
            }
        }
    }
    for (int j = 2; j <= m; j++) {
        // 再数横着相连的草
        for (int i = 1; i <= n; i++) {
            if (g[i][j] == '#' && g[i][j - 1] == '#') {
```

```
                sum++;
                g[i][j] = g[i][j - 1] = '.';
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (g[i][j] == '#')sum++;
        }
    }
    printf("%d\n", sum);
    return 0;
}
```

H

判断子串，因题目数据量小，可以使用下面的代码实现，但是此方法效率不高，有兴趣可以了解一下有名的 KMP 算法。

```
#include <stdio.h>
#include <string.h>

#define N 205
char str1[N], str2[N];

int main() {
    scanf("%s", str1);
    scanf("%s", str2);
    int len1 = strlen(str1);
    int len2 = strlen(str2);
    for (int i = 0; i + len1 - 1 < len2; i++) {
        for (int j = 0; str1[j] == str2[i + j]; j++) {
            if (j == len1 - 1) {
                printf("%s is substring of %s\n", str1, str2);
                return 0;
            }
        }
    }
    for (int i = 0; i + len2 - 1 < len1; i++) {
        for (int j = 0; str2[j] == str1[i + j]; j++) {
            if (j == len2 - 1) {
                printf("%s is substring of %s\n", str2, str1);
                return 0;
            }
        }
    }
    printf("No substring\n");
    return 0;
}
```

I

如果暴力枚举，时间复杂度 $O(n^2)$ 而 n 的数据量有 10^5 ，一定会超时。

应该先对数组进行排序（快速排序算法时间复杂度 $O(nlogn)$ 。本题解使用了 stdlib 中的库函数 qsort，有兴趣可以自行了解 qsort 函数用法及实现），再从小到大枚举第一个数 a[i]，因为此时排序后的数组具有单调性，可以通过二分查找来找到第二个数 m-a[i]，该过程时间复杂度 $O(nlogn)$。

```
#include <stdio.h>
#include <stdlib.h>

#define N 100005

int a[N];

int cmp(const void *a, const void *b) {
    return *(int *) a - *(int *) b; //由小到大排序
    //return *(int *)b - *(int *)a; 由大到小排序
}

int main() {
    int n, m;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)scanf("%d", &a[i]);
    scanf("%d", &m);
    //排序
    qsort(a, n, sizeof(int), cmp);
    for (int i = 0; i < n; i++) {
        int left = i + 1, right = n - 1;
        // 二分查找
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (a[mid] == m - a[i]) {
                printf("%d %d\n", a[i], a[mid]);
                return 0;
            } else if (a[mid] < m - a[i])
                left = mid + 1;
            else if (a[mid] > m - a[i])
                right = mid - 1;
        }
    }
    printf("No\n");
    return 0;
}
```