代码魔法-入门篇

变量的命名

```
int i, j, k, a, b, c, d, e;

            vs

int sum, result, new_student, good_person;
```

变量的作用域

```
int result = 1;
{
    int result = 0;
    cout << result << endl;
}
cout << result << endl;
```

```
//有什么好处
do {
    ...
    if (...) {
        break;
    }
    ...
} while(false);
```

常量的设置

```
const int N = 100010;
const int M = 100010;
const int MAX_MAGIC = 100010;

int nodes[N], edges[M], magic[MAX_MAGIC];
```

```
(a *= 2) %= 1000000007;
(a += 1) %= 1000000007;

const int MOD = 1000000007;

(a *= 2) %= MOD;
(a += 1) %= MOD;
```

不推荐使用 `#define` 定义常量

避免重复

```
  dp[x][y] = max(dp[x][y], dp[x - 1][y]);
  dp[x][y] = max(dp[x][y], dp[x - 1][y - 1] );
```

```
void update(int &x, int y) {
    if (y > x) {
        x = y;
    }
}

{
    update(dp[x][y], dp[x - 1][y]);
    update(dp[x][y], dp[x - 1][y - 1]);
}
```

结构体编程

```cpp
struct node {
    //定义数据结构
    int x, y;

    //构造函数，返回一个node对象
    node (int _x = 0, int _y = 0): x(_x), y(_y) {
    }

    //自定义小于号
    bool operator < (const node& cmp) const {
        return x < cmp.x || x == cmp.x && y < cmp.y;
    }

    void print () {
        cout << x << " " << y << endl;
    }
};

int main() {
    node a[10];
    for (int i = 0; i < 10; i++) {
        a[i] = node(10 - i, 10 - i); // 调用构造函数
    }
    sort(a, a + 10); //如果没有自定义小于号这一行会编译错误
    for (int i = 0; i < 10; i++) {
        a[i].print();
    }
    return 0;
}
```

```
struct problem{
    int n;
    int a[N];

    void input () {
    }

    void prepare() {
    }

    void solve() {
    }
};

int main() {
    problem solver;
    solver.input();
    solver.prepare();
    solver.solve();
    return 0;
}
```

命名空间

```cpp
#include <bits/stdc++.h>


int main()
{
    std::cout << std::max(1, 2) << std::endl;
    return 0;
}
```

```cpp
#include <bits/stdc++.h>

namespace solve {
    void say_hello() {
        std::cout << "hello" << std::endl;
    }
}

int main()
{
    solve::say_hello();
    return 0;
}
```

一个函数只完成一件事情

```
void combine(int l, int m, int r) {

}

void merge_sort(int l, int r) {
    if (l == r) {
        return ;
    }
    int m = l + r >> 1;
    merge_sort(l, m);
    merge_sort(m + 1, r);
    combine(l, m, r);
}
```

安全性

```c
while (scanf("%d", &n), n) {
}

while (scanf("%d", &n) != EOF && n) {
}

while (scanf("%d", &n) == 1 && n) {
}
```

正确性检验

检验正确性

Polya 定理：群作用下等价类数量是不动点数量的平均值

```
int sum = 0;
for (int i = 0; i < n; ++ i) {
    sum += count_fix_points(i);
}
assert(sum % n == 0);
printf("%d\n", sum / n);
```

对称性

统计 $(0,0)-(a,0)-(0,b)$ 内的格点数量

```
int solve(int a, int b) {
    int result = 0;
    for (int i = 1; i < a; ++ i) {
        result += b * i / a;
    }
    return result;
}

assert(solve(a, b) == solve(b, a));
```

获得对应返回

```
void runtime_error() {
    printf("%d\n", 1 / 0);
}
```

```
void time_limit_exceed() {
    int result = 1;
    while (true) {
        result <<= 1;
    }
    printf("%d\n", result);
}
```

代码风格

常量、宏 MAX_COUNT, MAGIC_NUMBER, FOREACH

类型 Pair, AvlTree

变量 pair, avlTree, makeLifeBetter

常量 MAX_COUNT, MAGIC_NUMBER, FOREACH

类型 Pair, AvlTree

变量、函数 pair, avl_tree, make_life_better

选择并坚持