241380022-沈悠然 的 CPL Homework DotOJ 2024 年度报告

> 数据统计截止时间：2025 年 1 月 1 日 0 时 15 分
>
> 本报告仅由作业 *OJ* (https://oj.cpl.icu) 上的数据进行统计，不包含机试 *OJ*。
>
> 多文件题的代码行数无法统计，因此不包含在内，请见谅。

与君初相识，你的 *OJ* 之旅从这里开始

初来乍到，你的第一次提交是在 2024 年 09 月 27 日 22：44：12，那时候你向 **世界** 问好了吗?

c

```c
#include <stdio.h>
int main() {
    printf("Hello C Programming!");
    return 0;
}
```

跋山涉水，你在 *OJ* 上的足迹

在这一学期里，你在 *OJ* 上奋战了 37 天，共提交了 458 次代码，其中有 179 次被评为 **答案正确**，正确提交占比达 39.08%。

你的提交中 **答案正确** 出现的次数最多，共有 179 次。

另外你还自定测试了 40 次，从中你发现了多少 *Bug* 呢?

你在这一学期里共参加了 24 次作业/练习，平均排名在 70 名。

排名最高的那次是 2024-exam-replica，你排到了第 2 名，你还记得当时的风采吗?

你一共尝试了 129 道题目，其中有 128 道题目成功被你解决，超过了 100.00% 的同学。

> 有道是天道酬勤，期末一定会考好的！

编程之山，你正在一步步攀登

在这一学期里，你一共提交了 36139 行代码，总共有 1049835 字节，平均每次提交 78.91 行代码，你代码的屎山程度超过了 87.31% 的同学!

> 回首自己这一学期所写下的这么多代码，你成长了多少呢？

你的代码总共在 *OJ* 上运行了 01 分钟 01 秒，平均每次提交运行 134.19 毫秒，你的代码**运行速度**超过了 38.78% 的同学。

> 一寸光阴一寸金，评测姬的时间你都好好把握住了吗？

你的代码总共消耗了 7.75 GB 内存，平均每次提交消耗 17749.73 KB 内存。

> 算算这些内存需要买多少内存条呢？

你的 *OJ* 之旅，有过这些难忘的瞬间

必做题（behaviors.c） 这道题你还记得吗？你尝试了 40 次，第 40 次终于被评为 **答案正确** 获得满分，**你的耐心和毅力真的值得称赞！**

c

```c
#include <stdio.h>
int n, a1 = -1, a2, a3, x, a[19];
int main() {
    scanf("%d", &n);
    a[3] = 3;
    a[14] = 14;
    a[15] = 15;
    a[16] = 16;
```

```c
    a[18] = 18;
    for (int i = 1, x; i <= n; i++) {
        scanf("%d", &x);
        a1 &= a[x];
        a2 |= a[x];
        a3 ^= a[x];
    }
    printf("%d %d %d", a1, a2, a3);
    return 0;
}
```

你在 命令行浏览器 这道题上提交了最长的代码 672186，共 648 行 19386 个字节，那天你是不是有些拉肚子?

c

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
const char ch_skip[] = "\n\r\t ";
char ans[11][51];
int X, COL[11][51];
bool EM[11][51], I[11][51], U[11][51], RESET[11][51];
enum type {
    _h, _p, _img, _div,
    _none = -1
} st[101];
enum color {
    red, blue, green,
    none = -1
};
```

```
enum direction {
    row, column
};
enum align {
    start, end, center, space_evenly
};
struct text {
    enum color c;
    bool em, i, u;
    char s[51];
};
struct image {
    char src[501];
    int width;
};
struct size {
    int w, h;
};
struct division {
    struct size siz;
    int cnt;
    enum direction dir;
    enum align ai, jc;
    enum type t[101];
    struct text txt[101];
    struct image img[101];
    struct division *div[101];
};
void init_text(struct text *txt) {
    txt -> c = none;
    txt -> em = false;
    txt -> i = false;
    txt -> u = false;
}
void init_division(struct division *d) {
    d -> siz.w = 0;
```

```
    d -> siz.h = 0;
    d -> cnt = 0;
    d -> dir = row;
    d -> ai = start;
    d -> jc = start;
    memset(d -> t, -1, sizeof(d -> t));
    for (int i = 0; i <= 100; i++) init_text(&d -> txt[i]
    memset(d -> img, 0, sizeof(d -> img));
    for (int i = 0; i <= 100; i++) d -> div[i] = NULL;
}
struct division *create() {
    struct division *d = (struct division*)malloc(sizeof(
    init_division(d);
    return d;
}
void write_p(struct text txt, int x, int y) {
    bool reset = false;
    if (txt.c != -1) {
        COL[x][y] = txt.c;
        reset = true;
    }
    if (txt.em) {
        EM[x][y] = true;
        reset = true;
    }
    if (txt.i) {
        I[x][y] = true;
        reset = true;
    }
    if (txt.u) {
        U[x][y] = true;
        reset = true;
    }
    int len = (int)strlen(txt.s);
    for (int j = 0; j < len; j++) ans[x][y+j] = txt.s[j];
    if (reset) RESET[x][y+len-1] = true;
```

```
}
void write_h(struct text txt, int x, int y) {
    for (int i = 0; txt.s[i]; i++) txt.s[i] = toupper(txt
    write_p(txt, x, y);
}
void write_img(struct image img, int x, int y) {
    for (int i = x, j = y, k = 0; img.src[k]; j++, k++) {
        if (j - y == img.width) {
            i++;
            j = y;
        }
        ans[i][j] = img.src[k];
    }
}
struct size siz_txt(struct text txt) {
    return (struct size){(int)strlen(txt.s), 1};
}
struct size siz_img(struct image img) {
    return (struct size){img.width, (int)strlen(img.src)
}
struct size siz_div(struct division *d) {
    int maxw = 0, maxh = 0, sumw = 0, sumh = 0;
    for (int i = 1; i <= d -> cnt; i++) {
        struct size cur;
        switch (d -> t[i]) {
            case _h: case _p: {
                cur = siz_txt(d -> txt[i]);
                break;
            }
            case _img: {
                cur = siz_img(d -> img[i]);
                break;
            }
            case _div: {
                cur = siz_div(d -> div[i]);
                break;
```

```
            }
            default: assert(false);
        }
        if (cur.w > maxw) maxw = cur.w;
        if (cur.h > maxh) maxh = cur.h;
        sumw += cur.w;
        sumh += cur.h;
    }
    struct size siz = (struct size){d -> dir == row ? ma
    if (d -> siz.w != 0) siz.w = d -> siz.w;
    if (d -> siz.h != 0) siz.h = d -> siz.h;
    return siz;
}
void write_div(struct division *d, int x, int y) {
    if (d -> siz.w == 0 || d -> siz.h == 0) d -> siz = si
    struct size sonsiz[101];
    for (int i = 1; i <= d -> cnt; i++) {
        switch (d -> t[i]) {
            case _h: case _p: {
                sonsiz[i] = siz_txt(d -> txt[i]);
                break;
            }
            case _img: {
                sonsiz[i] = siz_img(d -> img[i]);
                break;
            }
            case _div: {
                sonsiz[i] = siz_div(d -> div[i]);
                break;
            }
            default: assert(false);
        }
    }
    int sonx[101] = {}, sony[101] = {};
    switch (d -> dir) {
        case row: {
```

```
switch (d -> ai) {
    case start: {
        sonx[1] = x;
        for (int i = 2; i <= d -> cnt; i++) s
        break;
    }
    case end: {
        sonx[d->cnt] = x + d->siz.h - sonsiz[
        for (int i = d -> cnt - 1; i >= 1; i-
        break;
    }
    case center: {
        int sumh = 0;
        for (int i = 1; i <= d -> cnt; i++) s
        sonx[1] = x + (d -> siz.h - sumh) / 2
        for (int i = 2; i <= d -> cnt; i++) s
        break;
    }
    case space_evenly: {
        int sumh = 0;
        for (int i = 1; i <= d -> cnt; i++) s
        sonx[1] = x + (d -> siz.h - sumh) / (
        for (int i = 2; i <= d -> cnt; i++) s
        break;
    }
}
switch (d -> jc) {
    case start: {
        for (int i = 1; i <= d -> cnt; i++) s
        break;
    }
    case end: {
        for (int i = 1; i <= d -> cnt; i++) s
        break;
    }
    case center: case space_evenly: {
```

```
                for (int i = 1; i <= d -> cnt; i++) s
                break;
            }
        }
        break;
    }
    case column: {
        switch (d -> ai) {
            case start: {
                for (int i = 1; i <= d -> cnt; i++) s
                break;
            }
            case end: {
                for (int i = 1; i <= d -> cnt; i++) s
                break;
            }
            case center: case space_evenly: {
                for (int i = 1; i <= d -> cnt; i++) s
                break;
            }
        }
        switch (d -> jc) {
            case start: {
                sony[1] = y;
                for (int i = 2; i <= d -> cnt; i++) s
                break;
            }
            case end: {
                sony[d->cnt] = y + d->siz.w - sonsiz[
                for (int i = d -> cnt - 1; i >= 1; i-
                break;
            }
            case center: {
                int sumw = 0;
                for (int i = 1; i <= d -> cnt; i++) s
                sony[1] = y + (d -> siz.w - sumw) / 2
```

```
                for (int i = 2; i <= d -> cnt; i++) s
                break;
            }
            case space_evenly: {
                int sumw = 0;
                for (int i = 1; i <= d -> cnt; i++) s
                sony[1] = y + (d -> siz.w - sumw) / (
                for (int i = 2; i <= d -> cnt; i++) s
                break;
            }
        }
        break;
    }
    default: assert(false);
}
for (int i = 1; i <= d -> cnt; i++) {
    switch (d -> t[i]) {
        case _h: {
            write_h(d -> txt[i], sonx[i], sony[i]);
            break;
        }
        case _p: {
            write_p(d -> txt[i], sonx[i], sony[i]);
            break;
        }
        case _img: {
            write_img(d -> img[i], sonx[i], sony[i]);
            break;
        }
        case _div: {
            write_div(d -> div[i], sonx[i], sony[i]);
            break;
        }
        default: assert(false);
    }
}
```

```
}
char getch() {
    for (char ch = getchar(); ; ch = getchar()) {
        bool skip = false;
        for (int i = 0; ch_skip[i]; i++) {
            if (ch == ch_skip[i]) {
                skip = true;
                break;
            }
        }
        if (!skip) return ch;
    }
}
void skip_ch(int n) {
    for (int i = 0; i < n; i++) getch();
}
void read_till(char *s, char c) {
    int l = 0;
    while (true) {
        s[l++] = getchar();
        if (s[l-1] == c) {
            s[l-1] = '\0';
            break;
        }
    }
}
int read_int() {
    char s[3];
    read_till(s, '\"');
    int x = s[0] - '0';
    if (strlen(s) == 2) x = x * 10 + s[1] - '0';
    return x;
}
struct text read_txt(enum type t, enum color c, bool em,
    struct text txt;
    init_text(&txt);
```

```
        txt.c = c;
        txt.em = em;
        txt.i = i;
        txt.u = u;
        while (true) {
            char ch_p = getch();
            if (ch_p == '>') break;
            switch (ch_p) {
                case 'e': {
                    txt.em = true;
                    skip_ch(1);
                    break;
                }
                case 'i': {
                    txt.i = true;
                    break;
                }
                case 'u': {
                    txt.u = true;
                    break;
                }
                case 'c': {
                    skip_ch(6);
                    char ch_c = getch();
                    switch (ch_c) {
                        case 'r': {
                            txt.c = red;
                            skip_ch(3);
                            break;
                        }
                        case 'b': {
                            txt.c = blue;
                            skip_ch(4);
                            break;
                        }
                        case 'g': {
```

```
                        txt.c = green;
                        skip_ch(5);
                        break;
                    }
                }
                break;
            }
        }
    }
    read_till(txt.s, '<');
    skip_ch(3);
    return txt;
}
struct image read_img() {
    struct image img;
    while (true) {
        char ch_p = getch();
        if (ch_p == '>') break;
        switch (ch_p) {
            case 's': {
                skip_ch(4);
                read_till(img.src, '\"');
                break;
            }
            case 'w': {
                skip_ch(6);
                img.width = read_int();
                break;
            }
        }
    }
    skip_ch(6);
    return img;
}
struct division *read_div(enum color c, bool em, bool i,
    struct division *div = create();
```

```
while (true) {
    char ch_p = getch();
    if (ch_p == '>') break;
    switch (ch_p) {
        case 'e': {
            em = true;
            skip_ch(1);
            break;
        }
        case 'i': {
            i = true;
            break;
        }
        case 'u': {
            u = true;
            break;
        }
        case 'c': {
            skip_ch(6);
            char ch_c = getch();
            switch (ch_c) {
                case 'r': {
                    c = red;
                    skip_ch(3);
                    break;
                }
                case 'b': {
                    c = blue;
                    skip_ch(4);
                    break;
                }
                case 'g': {
                    c = green;
                    skip_ch(5);
                    break;
                }
```

```
            }
            break;
        }
        case 'w': {
            skip_ch(2);
            div -> siz.w = read_int();
            break;
        }
        case 'h': {
            skip_ch(2);
            div -> siz.h = read_int();
            break;
        }
        case 'd': {
            skip_ch(10);
            char ch_d = getch();
            switch (ch_d) {
                case 'r': {
                    div -> dir = row;
                    skip_ch(3);
                    break;
                }
                case 'c': {
                    div -> dir = column;
                    skip_ch(6);
                    break;
                }
                default: assert(false);
            }
            break;
        }
        case 'a': {
            skip_ch(12);
            char ch_a[2];
            ch_a[0] = getch();
            ch_a[1] = getch();
```

```
            switch (ch_a[0]) {
                case 's': {
                    switch (ch_a[1]) {
                        case 't': {
                            div -> ai = start;
                            skip_ch(4);
                            break;
                        }
                        case 'p': {
                            div -> ai = space_evenly;
                            skip_ch(11);
                            break;
                        }
                        default: assert(false);
                    }
                    break;
                }
                case 'c': {
                    div -> ai = center;
                    skip_ch(5);
                    break;
                }
                case 'e': {
                    div -> ai = end;
                    skip_ch(2);
                    break;
                }
                default: assert(false);
            }
            break;
        }
        case 'j': {
            skip_ch(16);
            char ch_j[2];
            ch_j[0] = getch();
            ch_j[1] = getch();
```

```
                switch (ch_j[0]) {
                    case 's': {
                        switch (ch_j[1]) {
                            case 't': {
                                div -> jc = start;
                                skip_ch(4);
                                break;
                            }
                            case 'p': {
                                div -> jc = space_evenly;
                                skip_ch(11);
                                break;
                            }
                            default: assert(false);
                        }
                        break;
                    }
                    case 'c': {
                        div -> jc = center;
                        skip_ch(5);
                        break;
                    }
                    case 'e': {
                        div -> jc = end;
                        skip_ch(2);
                        break;
                    }
                    default: assert(false);
                }
                break;
            }
        }
    }
    for (char ch = getch(); ch != EOF; ch = getch()) {
        if (ch != '<') break;
        char ch_t = getch();
```

```
enum type t = _none;
if (ch_t == '/') {
    skip_ch(4);
    return div;
}
switch (ch_t) {
    case 'h': {
        t = _h;
        break;
    }
    case 'p': {
        t = _p;
        break;
    }
    case 'i': {
        t = _img;
        skip_ch(2);
        break;
    }
    case 'd': {
        t = _div;
        skip_ch(2);
        break;
    }
    default: assert(false);
}
div -> t[++div->cnt] = t;
switch (t) {
    case _h: case _p: {
        div -> txt[div->cnt] = read_txt(t, c, em,
        break;
    }
    case _img: {
        div -> img[div->cnt] = read_img();
        break;
    }
```

```
                case _div: {
                    div -> div[div->cnt] = read_div(c, em, i,
                    break;
                }
                default: assert(false);
            }
        }
        skip_ch(5);
        return div;
    }
    int main() {
        memset(COL, -1, sizeof(COL));
        for (char ch = getch(); ch != EOF; ch = getch()) {
            assert(ch == '<');
            char ch_t = getch();
            enum type t = _none;
            switch (ch_t) {
                case 'h': {
                    t = _h;
                    break;
                }
                case 'p': {
                    t = _p;
                    break;
                }
                case 'i': {
                    t = _img;
                    skip_ch(2);
                    break;
                }
                case 'd': {
                    t = _div;
                    skip_ch(2);
                    break;
                }
                default: assert(false);
```

```
        }
        switch (t) {
            case _h: {
                struct text txt_h = read_txt(t, none, fa
                write_h(txt_h, X, 0);
                X++;
                break;
            }
            case _p: {
                struct text txt_p = read_txt(t, none, fa
                write_p(txt_p, X, 0);
                X++;
                break;
            }
            case _img: {
                struct image img = read_img();
                write_img(img, X, 0);
                X += strlen(img.src) / img.width;
                break;
            }
            case _div: {
                struct division *div = read_div(none, fa
                write_div(div, X, 0);
                X += div -> siz.h;
                break;
            }
            default: assert(false);
        }
    }
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 50; j++) {
            switch (COL[i][j]) {
                case red: {
                    printf("\033[31m");
                    break;
                }
```

```
                case blue: {
                    printf("\033[34m");
                    break;
                }
                case green: {
                    printf("\033[32m");
                    break;
                }
            }
            if (EM[i][j]) printf("\033[1m");
            if (I[i][j]) printf("\033[3m");
            if (U[i][j]) printf("\033[4m");
            putchar(ans[i][j] ? ans[i][j] : ' ');
            if (RESET[i][j]) printf("\033[0m");
        }
        putchar('\n');
    }
    return 0;
}
```

在 2023-exam-1-不计分模拟 这次作业/练习，你在 234 分钟内就解决了所有题目。

而在 说的道理 (reverse.c) 这道题你在 26 分钟内就获得了满分，你的速度真的很快！

在 2024 年 10 月 18 日 这一天你提交了 66 次代码，那时候你是不是想把 *OJ* 砸了呢？

这一学期以来，你挑灯夜战 *OJ* 共 3 天。

2024 年 12 月 17 日 00：38：14 你还在尝试提交 [冒泡排序 (bubble-sort.c)](#)……

> 写代码虽然有趣，可不要伤身哦！

[命令行浏览器](#)🖼️ 这题你尝试了 1 次却还没解决……

> 有时候学会放弃，也是一种智慧。

你的 *OJ* 之旅，还没结束……

期末机试加油！
