Standard Code Library

Your TeamName

Your School

December 16, 2021

Contents

编译原理实验 1

代码

- 需要 C++11

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

using namespace std;

enum {
    Comment = 0, Main, If, Then, While, Do, Static, Int, Double, Struct,
    Break, Else, Long, Switch, Case, Typedef, Char, Return, Const, Float,
    Short, Continue, For, Void, Sizeof, ID, NUM, ADD, SUB, MUL,
    DIV, COLON, DEFINE, LT, NE, LE, GT, GE, EQ, Default,
    Do1, SEMI, LB, RB
};

char name[][10] = {
        "Comment", "Main", "If", "Then", "While", "Do", "Static", "Int", "Double", "Struct",
        "Break", "Else", "Long", "Switch", "Case", "Typedef", "Char", "Return", "Const", "Float",
        "Short", "Continue", "For", "Void", "Sizeof", "ID", "NUM", "ADD", "SUB", "MUL",
        "DIV", "COLON", "DEFINE", "LT", "NE", "LE", "GT", "GE", "EQ", "Default",
        "Do1", "SEMI", "LB", "RB"
};


struct Token {
    int syn = 0;
    string token;
    int val = 0;

    void print() const {
        if (syn == -2) {
            printf("<%d,%s>\n", syn, token.c_str());
            return;
        }
        if (syn == NUM) {
            printf("<%d,", syn);
            for (int i = 31; i >= 0; --i) {
                printf("%d", (val >> i) & 1);
            }
            printf("b>\n");
        }
//        else if (syn == ID)
//            printf("<%d,%s>\n", syn, token.c_str());
        else
            printf("<%d,%s>\n", syn, token.c_str());
    }
};


int identifierNum = RB + 1;
unordered_map<string, int> identifier;

void initIdentifier() {
    identifier["#"] = Comment;
    identifier["main"] = Main;
    identifier["if"] = If;
    identifier["then"] = Then;
    identifier["while"] = While;
    identifier["do"] = Do;
    identifier["static"] = Static;
    identifier["int"] = Int;
    identifier["double"] = Double;
    identifier["struct"] = Struct;
    identifier["break"] = Break;
    identifier["else"] = Else;
    identifier["long"] = Long;
    identifier["switch"] = Switch;
    identifier["case"] = Case;
```

```
    identifier["typedef"] = Typedef;
    identifier["char"] = Char;
    identifier["return"] = Return;
    identifier["const"] = Const;
    identifier["float"] = Float;
    identifier["short"] = Short;
    identifier["continue"] = Continue;
    identifier["for"] = For;
    identifier["void"] = Void;
    identifier["sizeof"] = Sizeof;
    identifier["default"] = Default;

}

char *p, *lastp;
int line = 0;


Token *next() {
    auto *tk = new Token;
    tk->token = *p;
    while (*p) {//判断字符串是否结束
        if (*p == '\n') {
            printf("%d: ", line, (int) (p - lastp), lastp);
            while (lastp < p) putchar(*(lastp++));
            printf("\n\n");
            ++p;
            lastp = p;
            line = line + 1;
        } else if (isalpha(*p)) {
            tk->syn = ID;
            tk->token = "";
            while (isalpha(*p) || isdigit(*p)) {
                tk->token += *p;
                ++p;
            }
            if (identifier[tk->token] != 0) {
                if (identifier[tk->token] <= RB) {
                    tk->syn = identifier[tk->token];
                } else {
                    tk->val = identifier[tk->token];
                }
            } else {
                tk->val = identifier[tk->token] = ++identifierNum;
            }
            return tk;
        } else if (*p >= '0' && *p <= '9') {
            tk->syn = NUM;
            tk->val = 0;
            tk->token = "";
            while (*p >= '0' && *p <= '9' || isalpha(*p)) {
                if (isalpha(*p)) {
                    tk->syn = -2;
                } else {
                    tk->val = tk->val * 10 + *p - '0';
                }
                tk->token += *p;
                ++p;
            }
            return tk;
        } else {
            tk->token = *p;
            if (*p == '+') {
                ++p;
                tk->syn = ADD;
                return tk;
            } else if (*p == '-') {
                ++p;
                tk->syn = SUB;
                return tk;
            } else if (*p == '*') {
```

```
            ++p;
            tk->syn = MUL;
            return tk;
        } else if (*p == '/') {
            ++p;
            if (*p == '*') {
                while (true) {
                    ++p;
                    if (!*p) {
                        tk->syn = -1;
                        return tk;
                    }
                    while (*p && *p != '*')
                        ++p;
                    if (*p == '*' && *(p + 1) == '/') {
                        p += 2;
                        break;
                    }
                }
                continue;
            } else {
                tk->syn = DIV;
                return tk;
            }
        } else if (*p == ':') {
            ++p;
            if (*p == '=') {
                ++p;
                tk->token += *p;
                tk->syn = DEFINE;
            } else {
                tk->syn = COLON;
            }
            return tk;
        } else if (*p == '<') {
            ++p;
            if (*p == '>') {
                ++p;
                tk->token += *p;
                tk->syn = NE;
            } else if (*p == '=') {
                ++p;
                tk->token += *p;
                tk->syn = LE;
            } else {
                tk->syn = LT;
            }
            return tk;
        } else if (*p == '>') {
            ++p;
            if (*p == '=') {
                ++p;
                tk->token += *p;
                tk->syn = GE;
            } else {
                tk->syn = GT;
            }
            return tk;
        } else if (*p == '=') {
            ++p;
            tk->syn = EQ;
            return tk;
        } else if (*p == ';') {
            ++p;
            tk->syn = SEMI;
            return tk;
        } else if (*p == '(') {
            ++p;
            tk->syn = LB;
            return tk;
        } else if (*p == ')') {
```

```
                ++p;
                tk->syn = RB;
                return tk;
            } else {
                if (!isblank(*p)) {
                    tk->syn = -2;
                    tk->token = "";
                    if (*p && *p != '\n' && !isblank(*p)) {
                        tk->token += *p;
                        ++p;
                    }
                    return tk;
                }
                ++p;
            }
        }
    }
    printf("%d: ", line);
    while (lastp < p) putchar(*(lastp++));
    tk->syn = -1;
    return tk;
}

char s[1000000];

int main() {
    initIdentifier();
    FILE *f = fopen("1.txt", "r");
    fread(s, 1000000, 1000000, f);
    p = lastp = s;
    Token *tk = next();
    while (tk->syn != -1) {
        tk->print();
        tk = next();
    }
    return 0;
}
```

数据结构

ST 表

- 二维

```
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

数学

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$; 否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i\lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$; 否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时，$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$; 否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

图论

LCA

- 倍增

```cpp
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

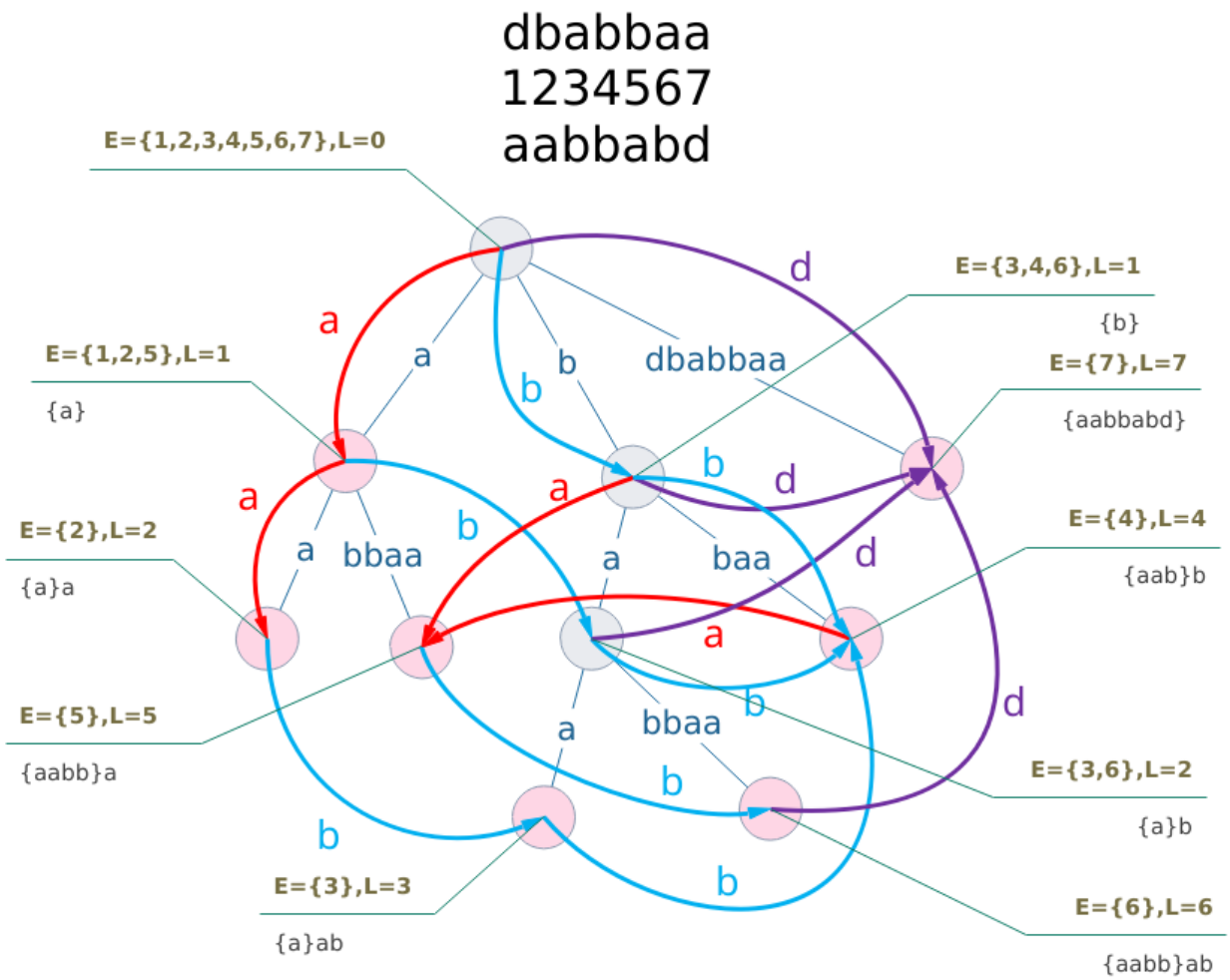
二维几何：点与向量

```cpp
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
```

```cpp
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// ---------------------------------------------
```

字符串

后缀自动机



杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```