

# Standard Code Library

Your TeamName

Your School

December 16, 2021

# Contents

<b>编译原理实验 1</b>	<b>2</b>
代码 . . . . .	2
<b>数据结构</b>	<b>5</b>
ST 表 . . . . .	5
<b>数学</b>	<b>6</b>
类欧几里得 . . . . .	6
<b>图论</b>	<b>6</b>
LCA . . . . .	6
<b>计算几何</b>	<b>6</b>
二维几何：点与向量 . . . . .	6
<b>字符串</b>	<b>7</b>
后缀自动机 . . . . .	7
<b>杂项</b>	<b>8</b>
STL . . . . .	8

## 编译原理实验 1

### 代码

- 需要 C++11

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  enum {
6      Comment = 0, Main, If, Then, While, Do, Static, Int, Double, Struct,
7      Break, Else, Long, Switch, Case, Typedef, Char, Return, Const, Float,
8      Short, Continue, For, Void, Sizeof, ID, NUM, ADD, SUB, MUL,
9      DIV, COLON, DEFINE, LT, NE, LE, GT, GE, EQ, Default,
10     Do1, SEMI, LB, RB
11 };
12
13 char name[][10] = {
14     "Comment", "Main", "If", "Then", "While", "Do", "Static", "Int", "Double", "Struct",
15     "Break", "Else", "Long", "Switch", "Case", "Typedef", "Char", "Return", "Const", "Float",
16     "Short", "Continue", "For", "Void", "Sizeof", "ID", "NUM", "ADD", "SUB", "MUL",
17     "DIV", "COLON", "DEFINE", "LT", "NE", "LE", "GT", "GE", "EQ", "Default",
18     "Do1", "SEMI", "LB", "RB"
19 };
20
21
22 struct Token {
23     int syn = 0;
24     string token;
25     int val = 0;
26
27     void print() const {
28         if (syn == -2) {
29             printf("<%d,%s>\n", syn, token.c_str());
30             return;
31         }
32         if (syn == NUM) {
33             printf("<%d,", syn);
34             for (int i = 31; i >= 0; --i) {
35                 printf("%d", (val >> i) & 1);
36             }
37             printf("b>\n");
38         }
39         // else if (syn == ID)
40         //     printf("<%d,%s>\n", syn, token.c_str());
41         else
42             printf("<%d,%s>\n", syn, token.c_str());
43     }
44 };
45
46
47 int identifierNum = RB + 1;
48 unordered_map<string, int> identifier;
49
50 void initIdentifier() {
51     identifier["#"] = Comment;
52     identifier["main"] = Main;
53     identifier["if"] = If;
54     identifier["then"] = Then;
55     identifier["while"] = While;
56     identifier["do"] = Do;
57     identifier["static"] = Static;
58     identifier["int"] = Int;
59     identifier["double"] = Double;
60     identifier["struct"] = Struct;
61     identifier["break"] = Break;
62     identifier["else"] = Else;
63     identifier["long"] = Long;
64     identifier["switch"] = Switch;
65     identifier["case"] = Case;
```

```

66     identifier["typedef"] = Typedef;
67     identifier["char"] = Char;
68     identifier["return"] = Return;
69     identifier["const"] = Const;
70     identifier["float"] = Float;
71     identifier["short"] = Short;
72     identifier["continue"] = Continue;
73     identifier["for"] = For;
74     identifier["void"] = Void;
75     identifier["sizeof"] = Sizeof;
76     identifier["default"] = Default;
77
78 }
79
80 char *p, *lastp;
81 int line = 0;
82
83
84 Token *next() {
85     auto *tk = new Token;
86     tk->token = *p;
87     while (*p) { //判断字符串是否结束
88         if (*p == '\n') {
89             printf("%d: ", line, (int) (p - lastp), lastp);
90             while (lastp < p) putchar(*(lastp++));
91             printf("\n\n");
92             ++p;
93             lastp = p;
94             line = line + 1;
95         } else if (isalpha(*p)) {
96             tk->syn = ID;
97             tk->token = "";
98             while (isalpha(*p) || isdigit(*p)) {
99                 tk->token += *p;
100                 ++p;
101             }
102             if (identifier[tk->token] != 0) {
103                 if (identifier[tk->token] <= RB) {
104                     tk->syn = identifier[tk->token];
105                 } else {
106                     tk->val = identifier[tk->token];
107                 }
108             } else {
109                 tk->val = identifier[tk->token] = ++identifierNum;
110             }
111             return tk;
112         } else if (*p >= '0' && *p <= '9') {
113             tk->syn = NUM;
114             tk->val = 0;
115             tk->token = "";
116             while (*p >= '0' && *p <= '9' || isalpha(*p)) {
117                 if (isalpha(*p)) {
118                     tk->syn = -2;
119                 } else {
120                     tk->val = tk->val * 10 + *p - '0';
121                 }
122                 tk->token += *p;
123                 ++p;
124             }
125             return tk;
126         } else {
127             tk->token = *p;
128             if (*p == '+') {
129                 ++p;
130                 tk->syn = ADD;
131                 return tk;
132             } else if (*p == '-') {
133                 ++p;
134                 tk->syn = SUB;
135                 return tk;
136             } else if (*p == '*') {

```

```

137     ++p;
138     tk->syn = MUL;
139     return tk;
140 } else if (*p == '/') {
141     ++p;
142     if (*p == '*') {
143         while (true) {
144             ++p;
145             if (!*p) {
146                 tk->syn = -1;
147                 return tk;
148             }
149             while (*p && *p != '*')
150                 ++p;
151             if (*p == '*' && *(p + 1) == '/') {
152                 p += 2;
153                 break;
154             }
155         }
156         continue;
157     } else {
158         tk->syn = DIV;
159         return tk;
160     }
161 } else if (*p == ':') {
162     ++p;
163     if (*p == '=') {
164         ++p;
165         tk->token += *p;
166         tk->syn = DEFINE;
167     } else {
168         tk->syn = COLON;
169     }
170     return tk;
171 } else if (*p == '<') {
172     ++p;
173     if (*p == '>') {
174         ++p;
175         tk->token += *p;
176         tk->syn = NE;
177     } else if (*p == '=') {
178         ++p;
179         tk->token += *p;
180         tk->syn = LE;
181     } else {
182         tk->syn = LT;
183     }
184     return tk;
185 } else if (*p == '>') {
186     ++p;
187     if (*p == '=') {
188         ++p;
189         tk->token += *p;
190         tk->syn = GE;
191     } else {
192         tk->syn = GT;
193     }
194     return tk;
195 } else if (*p == '=') {
196     ++p;
197     tk->syn = EQ;
198     return tk;
199 } else if (*p == ';') {
200     ++p;
201     tk->syn = SEMI;
202     return tk;
203 } else if (*p == '(') {
204     ++p;
205     tk->syn = LB;
206     return tk;
207 } else if (*p == ')') {

```

```

208         ++p;
209         tk->syn = RB;
210         return tk;
211     } else {
212         if (!isblank(*p)) {
213             tk->syn = -2;
214             tk->token = "";
215             if (*p && *p != '\n' && !isblank(*p)) {
216                 tk->token += *p;
217                 ++p;
218             }
219             return tk;
220         }
221         ++p;
222     }
223 }
224 }
225 printf("%d: ", line);
226 while (lastp < p) putchar(*(lastp++));
227 tk->syn = -1;
228 return tk;
229 }
230
231 char s[1000000];
232
233 int main() {
234     initIdentifier();
235     FILE *f = fopen("1.txt", "r");
236     fread(s, 1000000, 1000000, f);
237     p = lastp = s;
238     Token *tk = next();
239     while (tk->syn != -1) {
240         tk->print();
241         tk = next();
242     }
243     return 0;
244 }

```

## 数据结构

### ST 表

- 二维

```

1  int f[maxn][maxn][10][10];
2  inline int highbit(int x) { return 31 - __builtin_clz(x); }
3  inline int calc(int x, int y, int xx, int yy, int p, int q) {
4      return max(
5          max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
6          max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
7      );
8  }
9  void init() {
10     FOR (x, 0, highbit(n) + 1)
11     FOR (y, 0, highbit(m) + 1)
12     FOR (i, 0, n - (1 << x) + 1)
13     FOR (j, 0, m - (1 << y) + 1) {
14         if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
15         f[i][j][x][y] = calc(
16             i, j,
17             i + (1 << x) - 1, j + (1 << y) - 1,
18             max(x - 1, 0), max(y - 1, 0)
19         );
20     }
21 }
22 inline int get_max(int x, int y, int xx, int yy) {
23     return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
24 }

```

## 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

```
1 void dfs(int u, int fa) {
2     pa[u][0] = fa; dep[u] = dep[fa] + 1;
3     FOR (i, 1, SP) pa[u][i] = pa[pa[u][i-1]][i-1];
4     for (int& v: G[u]) {
5         if (v == fa) continue;
6         dfs(v, u);
7     }
8 }
9
10 int lca(int u, int v) {
11     if (dep[u] < dep[v]) swap(u, v);
12     int t = dep[u] - dep[v];
13     FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
14     FOR (i, SP-1, -1) {
15         int uu = pa[u][i], vv = pa[v][i];
16         if (uu != vv) { u = uu; v = vv; }
17     }
18     return u == v ? u : pa[u][0];
19 }
```

## 计算几何

### 二维几何：点与向量

```
1 #define y1 y1
2 #define nxt(i) ((i+1) % s.size())
3 typedef double LD;
4 const LD PI = 3.14159265358979323846;
5 const LD eps = 1E-10;
6 int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
7 struct L;
8 struct P;
9 typedef P V;
10 struct P {
11     LD x, y;
12     explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
13     explicit P(const L& l);
14 };
15 struct L {
16     P s, t;
17     L() {}
18     L(P s, P t): s(s), t(t) {}
19 };
20
21 P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
22 P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
```

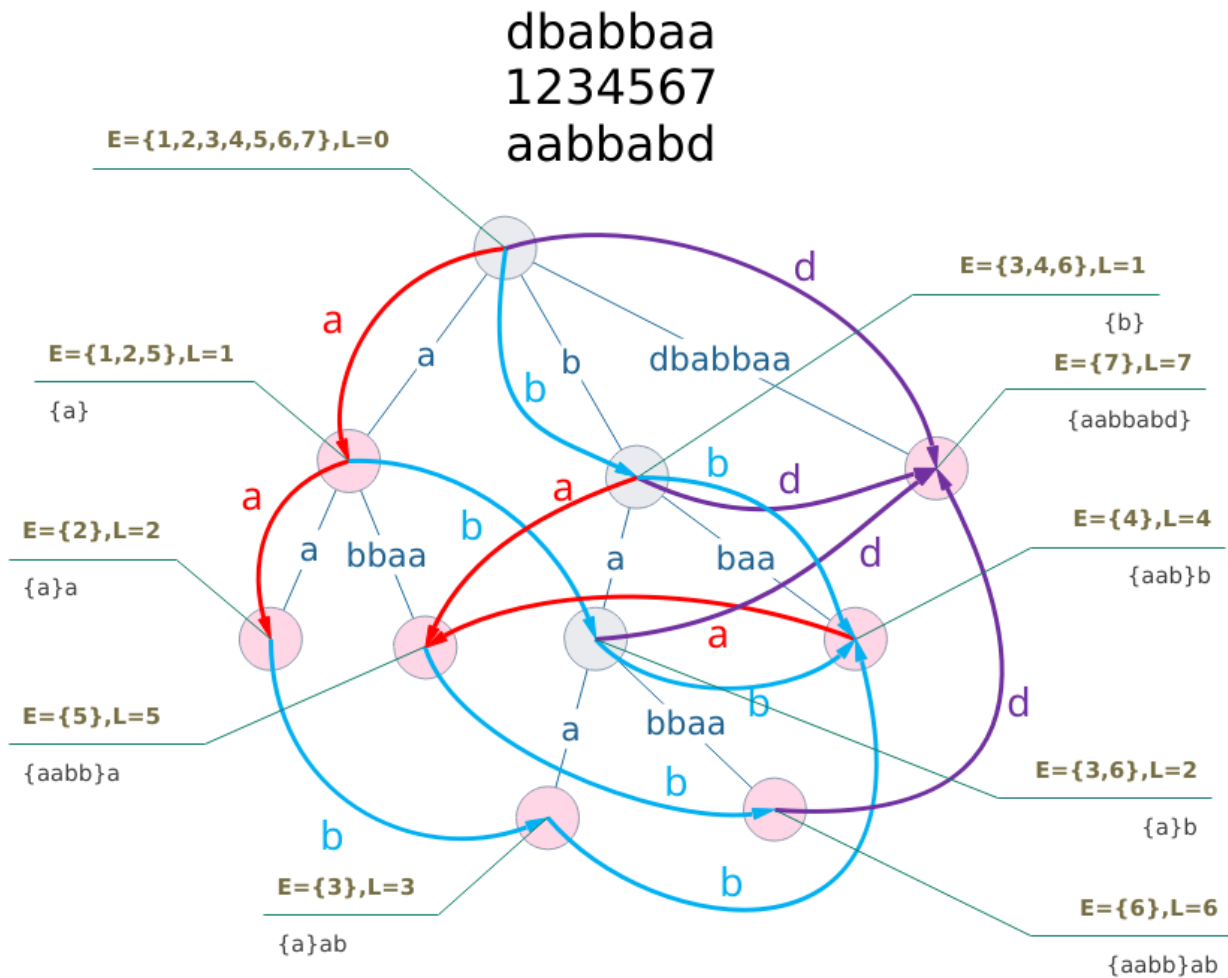
```

23 P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
24 P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
25 inline bool operator < (const P& a, const P& b) {
26     return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
27 }
28 bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
29 P::P(const L& l) { *this = l.t - l.s; }
30 ostream &operator << (ostream &os, const P &p) {
31     return (os << "(" << p.x << "," << p.y << ")");
32 }
33 istream &operator >> (istream &is, P &p) {
34     return (is >> p.x >> p.y);
35 }
36
37 LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
38 LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
39 LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
40 LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
41 // -----

```

## 字符串

### 后缀自动机





## 杂项

### STL

- copy

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