**/\*C program Lexikální a Syntaktická analýza PL0 – upozorníme jen na hlavní části \*/**

**#define NSYM 35 /\* pocet rozpoznatelnych symbolu\*/**

**#define NORW 11 /\* pocet klicovych slov \*/**

**#define TMAX 100 /\* velikost tabulky symbolu \*/**

**#define NMAX 5 /\* maximalni pocet cislic v cisle \*/**

**#define AL 10 /\* delka identifikatoru \*/**

**#define CHSETSIZE 128 /\* pocet znaku v mnozine \*/ *definice konstant***

**#define MAXERR 30 /\* maximalni pocet chyb \*/ *překladače***

**#define AMAX 2048 /\* nejvyssi adresa \*/**

**#define LEVMAX 3 /\* maximalni hloubka vnoreni \*/**

**#define CXMAX 200 /\* velikost prostoru pro kod \*/**

**#define STACKSIZE 500 /\* vypoctovy zásobník\*/**

**typedef enum {null, ident, number, plus, minus, times, slash, modulo, oddsym, eql,**

 **neq, lss, leq, gtr, geq, lparen, rparen, comma, semicolon, period,**

 **becomes, beginsym, endsym, ifsym, thensym, whilesym, dosym, callsym, *lex.symboly***

 **constsym, varsym, procsym} SYMBOL;**

**typedef char ALFA[AL]; /\*pole k ulozeni textu identifikatoru\*/**

**typedef enum {constant, variable, procedure} OOBJECT; /\*druh identifikatoru\*/**

**typedef int SYMSET[NSYM];**

**char ch; /\*posledni precteny znak\*/**

**FILE \*iva, \*zdroj; /\*pomocny soubor pro vypis generovaneho kodu\*/**

 **/\*a tabulky symbolu\*/**

**int txpom; /\*pomocna promenna\*/**

**SYMBOL sym; /\*posledni precteny symbol\*/**

**ALFA id; /\*posledni precteny identifikator\*/**

**int num; /\*posledni prectene cislo\*/**

**int cc; /\*pocet znaku\*/**

**int ll; /\*delka radku\*/**

**int kk,err;**

**int cx; /\*pocitadlo adres; vice bude receno v kap. Pridelovani pameti \*/**

**char line[81]; /\*nacteny radek\*/**

**ALFA a;**

**ALFA word[NORW]={"begin", "call", "const", "do", "end", "if", "odd", "procedure",**

 **"then", "var", "while"}; /\**pole rezervovaných identifikátorů\*/***

**SYMBOL wsym[NORW]={beginsym, callsym, constsym, dosym, endsym, ifsym, oddsym,**

 **procsym, thensym, varsym, whilesym}; /\*poradi převede na vyctový typ\*/**

**SYMBOL ssym[255]; /\*+,-,\*,…, jednoznakove oddělovače, plneno v main\*/**

**SYMSET declbegsys, statbegsys, facbegsys; /\*pocatecni symboly kostrukci, naplni je main \*/**

**struct {**

 **ALFA name; /\*jmeno\*/**

 **OOBJECT kind; /\*druh\*/**

 **union {**

 **int val;**

 **struct { *atributy identifikatoru Tabulka symbolů***

 **int level,adr,size;**

 **} vp;**

 **} CO;**

**} TABLE[TMAX+1];**

**/\* nacita ze vstupniho souboru 1 znak do glob. promenne 'ch' a prekroci konec radky \*/**

**void getch(void) { /\*sklada znaky do pole line\*/**

 **if (cc == ll) {**

 **if (feof(zdroj)) {**

 **printf("program incompleted");**

 **exit(2);**

 **}**

 **ll = cc = 0;**

 **printf(" ");**

 **do {**

 **fscanf(zdroj,"%c",&ch);**

 **if ((ch != '\n') && (ch != '\r')) {**

 **line[ll++]=ch; /\*pridani znaku do promenne line \*/**

 **printf("%c",ch);**

 **}**

 **} while ((ch != '\n') && (ch != '\r') && (feof(zdroj) == 0));**

 **printf("\n");**

 **line[ll++] = ' ';**

 **}**

 **ch = line[cc++];**

**} // getch()**

**/\* Lex.analyza nacita ze vstupniho souboru 1 symbol a vrati jeho kod do glob. promenne 'sym' \*/**

**void getsym(void) {**

**int i, j, k;**

 **while (ch <= ' ') getch(); /\* netisknutelne znaky \*/**

 **if ((ch >= 'a') && (ch <= 'z')) { /\* identifier or reserved word \*/**

 **k = 0;**

 **do { /\*identifikator\*/**

 **if (k < AL) a[k++] = ch;**

 **getch();**

 **} while (((ch >= 'a') && (ch <= 'z')) || ((ch >= '0') && (ch <= '9')));**

 **a[k] = '\0';**

 **strcpy(id, a);**

 **i = 0;**

 **j = NORW - 1;**

 **do {**

 **k = (i + j) / 2;**

 **if (strcmp(id, word[k]) <= 0) j = k - 1; /\**pulenim intervalu hleda v poli word\*/ /\*zda to je rezervovany identifikátor \*/***

 **if (strcmp(id, word[k]) >= 0) i = k + 1;**

 **} while (i <= j);**

 **if ((i - 1) > j) sym = wsym[k];**

 **else sym = ident;**

 **}**

 **else**

 **if ((ch >= '0') && (ch <= '9')) { /\*nasel začátek number \*/**

 **k = num = 0;**

 **sym = number;**

 **/\* vypusti pripadne pocatecni nuly u cisla \*/**

 **/\* while (ch == '0') getch(); \*/**

 **do {**

 **num = 10 \* num + (ch - '0'); /\*vypocet hodnoty cisla \*/**

 **k++;**

 **getch();**

 **} while ((ch >= '0') && (ch <= '9'));**

 **if (k > NMAX) error(30);**

 **} /\*konec number \*/**

 **else**

 **if (ch == ':') {**

 **getch();**

 **if (ch == '=') { prirazeni**

 **sym = becomes;**

 **getch();**

 **}**

 **else sym = null;**

 **}**

 **else**

 **if (ch == '<') {**

 **getch();**

 **if (ch == '=') { /\*mensi roven\*/**

 **sym = leq;**

 **getch();**

 **}**

 **else**

 **if (ch == '>') {**

 **sym = neq; /\*neroven\*/**

 **getch();**

 **}**

 **else**

 **sym = lss; /\*mensi\*/**

 **}**

 **else**

 **if (ch == '>') {**

 **getch();**

 **if (ch == '=') { /\*vetsi roven\*/**

 **sym = geq;**

 **getch();**

 **}**

 **else sym = gtr;**

 **}**

 **else {**

 **sym = ssym[ch]; /\*jednoznakovy oddělovač viz v main\*/**

 **getch();**

 **}**

**} /\* konec procedury getsym \*/**

**/\* vlozi object do tabulky symbolu**

 **k :typ objektu, tj. zda jde o konstantu,promennou,...**

 **lev :uroven, ve ktere je objekt deklarovan**

 **dx :adresa objektu**

**\*/**

**void enter(OOBJECT k,int \*tx,int lev,int \*dx) {**

 **(\*tx)++; /\* inkrementuje index tabulky symbolu \*/**

 **txpom = \*tx; /\* pro vypis TS \*/**

 **strcpy(TABLE[\*tx].name,id);**

 **TABLE[\*tx].kind=k;**

 **switch (k) {**

 **case constant: if (num>AMAX) {**

 **error(31);**

 **num = 0;**

 **}**

 **TABLE[\*tx].CO.val = num;**

 **break; /\*plneni tab. symbolu\*/**

 **case variable: TABLE[\*tx].CO.vp.level = lev;**

 **TABLE[\*tx].CO.vp.adr = (\*dx)++;**

 **break;**

 **case procedure: TABLE[\*tx].CO.vp.level = lev;**

 **break;**

 **}**

**} // enter()**

**/\* vyhleda symbol v tabulce symbolu**

 **id :jmeno symbolu**

 **tx :ukazovatko na konec tabulky symbolu**

 **navratova hodnota:**

 **-1 : symbol nenalezen**

 **>=0 : adresa symbolu**

**\*/**

**int position(ALFA id,int tx) {**

**int i;**

 **strcpy(TABLE[0].name,id); /\*tzv. sentinel\*/**

 **i = tx; /\*hleda od posledního zarazeneho (respektuje lokalitu\*/**

 **while (strcmp(TABLE[i].name,id)) i--;**

**return(i);**

**} // position()**

**/\* zpracovani deklarace konstanty ve tvaru:**

 **ident = hodnota.**

 **tx :ukazovatko na volne misto v tabulce symbolu**

 **lev :uroven, ve ktere je symbol deklarovan**

 **dx :adresa - neni pouzita, protoze jde tady o kontantu**

**\*/**

**void constdeclaration(int \*tx,int lev,int \*dx) {**

**if (sym == ident) {**

 **getsym();**

 **if ((sym == eql) || (sym == becomes)) {**

 **if (sym == becomes) error(1); /\*v deklaraci konstant musí byt „=“ \*/**

 **getsym();**

 **if (sym == number) {**

 **enter(constant,tx,lev,dx); /\*ulozeni konstanty do Tab.Symb, \*/**

 **getsym();**

 **} else error(2); /\*konstante není prirazeno cislo\*/**

 **} else error(3); /\*nenasel = ani prirazeni\*/**

**} else error(4); /\*nenasel identifikátor\*/**

**} // constdeclaration()**

**/\* zpracovani deklarace promenne**

 **tx :ukazovatko na volne misto v tabulce symbolu**

 **lev :uroven,ve ktere je symbol deklarovan**

 **dx :adresa promenne**

**\*/**

**void vardeclaration(int \*tx,int lev,int \*dx) {**

**if (sym == ident) {**

 **enter(variable,tx,lev,dx);**

 **getsym();**

**} else error(4);**

**} // vardeclaration()**

**void factor(…) { /\*v kompletnim tvaru bude mit parametry\*/**

**int i;**

 **while (facbegsys[sym]) { /\* facbegsys se naplni v main\*/**

 **if (sym == ident) {**

 **i = position(id,tx);**

 **if (i == 0) error(11); /\*nenalezen v Tab.Symb.\*/**

 **else**

 **getsym();**

 **} else**

 **if (sym == number) {**

 **if (num > AMAX) {**

 **error(31);**

 **num = 0;**

 **}**

 **getsym();**

 **} else**

 **if (sym == lparen) { getsym();**

 **expression(…);**

 **if (sym == rparen) getsym();**

 **else error(22);**

 **}**

 **}**

**} // factor()**

**void term(…) { /\*v kompletnim tvaru bude mit parametry\*/**

**SYMBOL mulop;**

 **factor(…);**

 **while ((sym == times) || (sym == slash) || (sym == modulo)) {**

 **mulop = sym;**

 **getsym();**

 **factor(…);**

 **}**

**} // term()**

**void expression(…) { /\*v kompletnim tvaru bude mit parametry\*/**

**SYMBOL addop;**

 **if ((sym == plus) || (sym == minus)) { /\*unární operatory\*/**

 **getsym();**

 **term(…);**

 **}**

 **else {**

 **term(…);**

 **}**

 **while ((sym == plus) || (sym == minus)) { /\*binární operatory\*/**

 **getsym();**

 **term(…);**

 **}**

**} // expression()**

**void condition(…) {**

**SYMBOL relop;**

 **if (sym == oddsym) {**

 **getsym();**

 **expression(…);**

 **}**

 **else {**

 **expression(…);**

 **if ((sym != eql) && (sym != neq) && (sym != lss) && (sym != gtr) && (sym != leq) &&**

 **(sym != geq))**

 **error(22);**

 **else {**

 **getsym();**

 **expression(…);**

 **}**

 **}**

**} // condition()**

**void statement(…) { /\*v kompletnim tvaru bude mit parametry\*/**

**int i;**

 **if (sym != ident) {**

 **error(10);**

 **do**

 **getsym();**

 **while (fsys[sym] == 0);**

 **}**

 **if (sym == ident) { /\*nalezen prikaz prirazeni\*/**

 **i = position(id,tx);**

 **if (i == 0) error(11); /\*nenasel se identifikátor\*/**

 **else**

 **if (TABLE[i].kind!=variable) { /\*prirazeni do jineho ident. nez promenna\*/**

 **error(12);**

 **i = 0;**

 **}**

 **getsym();**

 **if (sym == becomes) getsym();**

 **else error(13);**

 **expression(…);**

 **}**

 **else**

 **if (sym == callsym) {/\*nalezeno volani podprogramu\*/**

 **getsym();**

 **if (sym != ident) error(14);**

 **else {**

 **if ((i = position(id,tx)) == 0) error(11);**

 **else {**

 **if (TABLE[i].kind == procedure) gen(cal, …**

 **else error(15);**

 **}**

 **getsym();**

 **}**

 **}**

 **else**

 **if (sym == ifsym) { /\*podmineny prikaz\*/**

 **getsym();**

 **condition(…);**

 **if (sym == thensym) getsym();**

 **else error(16);**

 **statement(…);**

 **}**

 **else**

 **if (sym == beginsym) { /\*zacina novy blok\*/**

 **getsym();**

 **statement(…);**

 **while (sym == semicolon) {**

 **getsym();**

 **else error(10);**

 **statement(…);**

 **}**

 **if (sym == endsym) getsym(); /\*konci predchozi blok\*/**

 **else error(17);**

 **}**

 **else**

 **if (sym == whilesym) { /\*zacina cyklus while\*/**

 **condition(…);**

 **if (sym == dosym) getsym();**

 **else error(18);**

 **statement(…);**

 **}**

**} // statement()**

**void block(…) { /\*v kompletnim tvaru bude mit parametry\*/**

 **do {**

 **if (sym == constsym) { /\*deklaracni cast konstant\*/**

 **getsym();**

 **do {**

 **constdeclaration(…);**

 **while (sym == comma) {**

 **getsym();**

 **constdeclaration(…);**

 **}**

 **if (sym == semicolon) getsym();**

 **else error(5);**

 **} while (sym==ident);**

 **}**

 **if (sym == varsym) { /\*deklaracni cast promennych\*/**

 **getsym();**

 **vardeclaration(…);**

 **while (sym == comma) {**

 **getsym();**

 **vardeclaration(…);**

 **}**

 **if (sym == semicolon) getsym();**

 **else error(5);**

 **}**

 **while (sym == procsym) { /\*definice podprogramu\*/**

 **getsym();**

 **if (sym == ident) {**

 **enter(procedure);**

 **getsym();**

 **} else error(4);**

 **if (sym==semicolon) getsym();**

 **else error(5);**

 **block(…);**

 **if (sym == semicolon) {**

 **getsym();**

 **} else error(5);**

 **}**

 **} while (declbegsys[sym]); /\* declbegsys se plni v main\*/**

 **statement(…);**

**} // block()**

**/\*hlavni program\*/**

**main(void) {**

**char zdrojak[13];**

 **/\* cte jmeno souboru, dokud uzivatel nezada nenulovy retezec \*/**

 **do {**

 **printf("Zadej jmeno souboru obsahujiciho zdrojovy text: ");**

 **scanf("%s",zdrojak);**

 **} while (strlen(zdrojak) < 1);**

 **if ((iva = fopen("TAB.SYM", "w")) == NULL) {**

 **printf("\nCHYBA! Nepodarilo se otevrit soubor pro zapis tabulky symbolu...\n");**

 **return(-1);**

 **}**

 **/\* ...a pak otestuje, jestli soubor existuje \*/**

 **if ((zdroj = fopen(zdrojak, "r")) == NULL) {**

 **printf("\nCHYBA! Nepodarilo se otevrit soubor se zdrojovym textem [%s]...\n",zdrojak);**

 **return(-1);**

 **}**

 **for (ch=' ';ch<='\_';ch++) ssym[ch] = null;**

 **ssym['+'] = plus;**

 **ssym['-'] = minus;**

 **ssym['\*'] = times;**

 **ssym['/'] = slash;**

 **ssym['%'] = modulo;**

 **ssym['('] = lparen;**

 **ssym[')'] = rparen;**

 **ssym['='] = eql; /\*naplneni hodnot jednoznakových oddelovacu\*/**

 **ssym[','] = comma; /\*ostatním znakum zustane hodnota ssym = null\*/**

 **ssym['.'] = period;**

 **ssym['#'] = neq;**

 **ssym['<'] = lss;**

 **ssym['>'] = gtr;**

 **ssym[';'] = semicolon;**

 **nuluj(declbegsys);**

 **nuluj(statbegsys);**

 **nuluj(facbegsys);**

 **/\*v deklaracni casti se musi zacinat bud 'const','var' nebo 'procedure'\*/**

 **declbegsys[constsym] = declbegsys[varsym] = declbegsys[procsym] = 1;**

 **/\*ve statementu se musi zacinat bud 'begin','call','if','while' nebo ident.\*/**

 **statbegsys[beginsym] = statbegsys[callsym] = statbegsys[ifsym] = statbegsys[whilesym] = 1;**

 **/\*faktor muze byt bud ident., cislo nebo leva zavorka\*/**

 **facbegsys[ident] = facbegsys[number] = facbegsys[lparen] = 1;**

 **ch = ' ';**

 **kk = AL;**

 **getsym();**

 **block(…); /\*zavola preklad programu\*/**

 **if (sym != period) error(9); /\*a konci teckou\*/**

 **listtabsym();**

 **if (err == 0) {**

 **printf("\nno error in PL/0 program\n");**

 **}**

 **else printf("\n %2d error(s) in PL/0 program",err);**

**return(0);**

**}**