$ cat head

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Лабораторная работа №24 \*

\* \*

\* Преобразование арифметических \*

\* выражений \*

\* \*

\* Работу выполнил: \*

\* Стеснягин Семён \*

\* Группа: 80-102Б \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$ cat lab24\_var14.cpp

#include<stdio.h>

int i;

char ch;

struct node {

char value;

node \*left\_node;

node \*right\_node;

};

node \*tree;

void PrintTree(node \*current\_node)

{

static int level = 0;

level++;

if (current\_node)

{

PrintTree(current\_node->right\_node);

for (int i = 0; i < level; i++)

{

printf(" ");

}

printf("\\\_\_%c\n", current\_node->value);

PrintTree(current\_node->left\_node);

}

level--;

}

int isAlphaNumber()

{

return ((ch >= 'a') && (ch <= 'z')) || ((ch >= '0') && (ch <= '9'));

}

int isNumber(char c)

{

return (c >= '0') && (c <= '9');

}

int isZero(char c)

{

return c == '0';

}

node \*CreateNode(char c, node \*l, node \*r) {

node \*t = new node;

t->value = c;

t->left\_node = l;

t->right\_node = r;

return t;

}

node \*ExpressionToTree();

node \*fact()

{

node \*t;

scanf("%c", &ch);

if (ch == '(') {

t = ExpressionToTree();

if (ch != ')')

printf("ERROR: not )\n");

}

else if (isAlphaNumber()) t = CreateNode(ch, 0, 0);

else printf("ERROR: not alpha or number\n");

return t;

}

node \*term() {

node \*tm;

int done;

char ch1;

tm = fact();

done = 0;

while ((ch != '\n') && (!done)) {

scanf("%c", &ch);

if ((ch == '\*') || (ch == '/')) {

ch1 = ch;

tm = CreateNode(ch1, tm, fact());

}

else done = 1;

}

return tm;

}

node \*ExpressionToTree()

{

node \*ex;

int done;

char ch1;

ex = term();

done = 0;

while ((ch != '\n') && (!done)) {

if ((ch == '+') || (ch == '-')) {

ch1 = ch;

ex = CreateNode(ch1, ex, term());

}

else done = 1;

}

return ex;

}

void TreeToExpression(node \*tree) {

if (tree) {

if ((tree->value == '+') || (tree->value == '-')) printf("(");

TreeToExpression(tree->left\_node);

printf("%c", tree->value);

TreeToExpression(tree->right\_node);

if ((tree->value == '+') || (tree->value == '-')) printf(")");

}

}

void transtree(node \*tree) {

char cl, cr;

if (tree) {

if (tree->value == '+')

{

cl = tree->left\_node->value;

cr = tree->right\_node->value;

if (isNumber(cl) && isZero(cr))

{

tree->value = cl;

tree->left\_node = 0;

tree->right\_node = 0;

i = 1;

}

else if (isZero(cl) && isNumber(cr))

{

tree->value = cr;

tree->left\_node = 0;

tree->right\_node = 0;

i = 1;

}

else if (isZero(cl)) {

\*tree = \*tree->right\_node;

}

else if (isZero(cr)) {

\*tree = \*tree->left\_node;

}

}

transtree(tree->left\_node);

transtree(tree->right\_node);

}

}

int main() {

printf("Input expression:\n");

tree = ExpressionToTree();

PrintTree(tree);

printf("\n\n-----------------------\n\n");

TreeToExpression(tree);

i = 1;

while (i) {

i = 0;

transtree(tree);

}

printf("\n\n-----------------------\n\n");

PrintTree(tree);

printf("\n\n-----------------------\n\n");

TreeToExpression(tree);

printf("\n\n-----------------------\n\n");

return 0;

}

$ c++ lab24\_var14.cpp

$ ./a.out

Input expression:

0+(1+2)

\\_\_2

\\_\_+

\\_\_1

\\_\_+

\\_\_0

-----------------------

(0+(1+2))

-----------------------

\\_\_2

\\_\_+

\\_\_1

-----------------------

(1+2)

-----------------------

$ ./a.out

Input expression:

(2\*2)+(3-3)+0+2+(0+0)

\\_\_0

\\_\_+

\\_\_0

\\_\_+

\\_\_2

\\_\_+

\\_\_0

\\_\_+

\\_\_3

\\_\_-

\\_\_3

\\_\_+

\\_\_2

\\_\_\*

\\_\_2

-----------------------

((((2\*2+(3-3))+0)+2)+(0+0))

-----------------------

\\_\_2

\\_\_+

\\_\_3

\\_\_-

\\_\_3

\\_\_+

\\_\_2

\\_\_\*

\\_\_2

-----------------------

((2\*2+(3-3))+2)

-----------------------

$ ./a.out

Input expression:

2+3/3+5\*3-2

\\_\_2

\\_\_-

\\_\_3

\\_\_\*

\\_\_5

\\_\_+

\\_\_3

\\_\_/

\\_\_3

\\_\_+

\\_\_2

-----------------------

(((2+3/3)+5\*3)-2)

-----------------------

\\_\_2

\\_\_-

\\_\_3

\\_\_\*

\\_\_5

\\_\_+

\\_\_3

\\_\_/

\\_\_3

\\_\_+

\\_\_2

-----------------------

(((2+3/3)+5\*3)-2)

-----------------------