

```
/*
 *
 * Programm zum Erzeugen der Postfixform aus der Infixform
 * UPN.C (C) by Daniel Becker 1999
 * Letzte Änderung am 19.4.99
 *
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define STKMAX 256

/* ; = + - * / ^ ( ) */
unsigned char action[10][10]={/*0*/ {0,0,0,0,0,0,0,0,0,0},
/*;*/ {0,4,2,5,5,5,5,5,5,5},
/*=*/ {0,1,2,2,2,2,2,2,2,5},
/*+*/ {0,1,5,1,1,2,2,2,2,1},
/*-*/ {0,1,5,1,1,2,2,2,2,1},
/****/ {0,1,5,1,1,1,1,2,2,1},
/**/ {0,1,5,1,1,1,1,2,2,1},
/*^*/ {0,1,5,1,1,1,1,1,2,1},
/*(*/ {0,5,5,2,2,2,2,2,2,3},
/*)*/ {0,5,5,5,5,5,5,5,5,5}};

unsigned char translat[]=";+-*/^()", upn_str[100], stk[STKMAX];
int stk_count;

void error(char *s)
{
    printf("%s\n\n",s);
    exit(3);
}

/* nr gibt die Nummer des Operators zurück oder aber 0, wenns keiner ist.*/
int nr(unsigned char t)
{
    char *s,i;

    for (s=translat,i=1; *s; s++,i++)
        if (*s==t) return(i);
    return(0);
}

void push(unsigned char c) /* Ein Zeichen auf den Stack */
{
    if (stk_count>STKMAX)
        error("Stack overflow.");
    stk[stk_count++]=c;
}

unsigned char pop() /* Ein Zeichen vom Stack */
{
    if (stk_count==0)
        error("Stack underflow.");
    return(stk[--stk_count]);
}

unsigned char topstk() /* Oberstes Zeichen ausgeben */
{
    return(stk[stk_count-1]);
    /* auch möglich : #define topstk() (stk[stk_count-1]) */
}

int stk_empty() /* Eine "boolsche" Funktion */
{

```

```

return(!stk_count);
/* auch möglich : #define stk_empty() (!stk_count) */
}

char *upn(char *str)
{
    char *s,*e;
    unsigned char OprNr,nochmal;

    stk_count=0; push(';'); /* Stack initialisieren */
    s=str; e=upn_str; /* upn_str ist global */

    while (*s) /* Term ist Fall 4 - oder Stringende */
    {
        OprNr=nr(*s);
        do
        {
            nochmal=0; /* Wiederholung bei Fall 1 : 1 */
                       /* Abbruch bei Fall 4 und 5: 2 */

            switch(action[nr(topstk())][OprNr])
            {
                case 0: *(e++)=*(s++); break; /* leere Operation */
                case 1: *(e++)=pop(); nochmal=1; break;
                case 2: push(*(s++)); break;
                case 3: pop(); s++; break;
                case 4: while (!stk_empty()) *(e++)=pop(); /* Stackreste leeren */
                       s++; break; /* (*s) sollte jetzt NULL sein... */
                case 5: nochmal=2; error("Syntaxfehler"); break;
            }
            if (nochmal==2) break;
        }
        while (nochmal);
        if (nochmal==2) break; /* Wirklich abbrechen...*/
    }
    *e=0; /* Ergebnisstring terminieren */
    return(upn_str);
}

int syntax_not_ok(char *Postfix_str)
{
    char *s;
    int sigma;

    s=Postfix_str; sigma=0;
    while (*s && *s!=';') /* Diesermal ohne Semikolon */
    {
        /* strchr sucht nach dem Zeichen (*s) in dem String (translat) */
        /* Zeichen nicht in translat -> nicht Operator */

        if (!strchr(translat,*s))
            sigma++; /* Operand, da es keine Funktionen gibt */
        else
            sigma--; /* Operator */
        if (!sigma) return(1); /* Nicht ok */

        s++;
    }
    if (!(*s)) return(2); /* Semikolon vergessen! tse,tse,tse... */
    if (sigma!=1) return(1); /* (all-)gemeiner Fehler... */
    return(0); /* Wenn's bis hier geht, dann ist's ok... */
}

void assemble(char *Postfix_str)

```

```

{
    char *s;

    s=Postfix_str;

    while (*s) /* Nullterminierung von Strings */
    {
        if (!nr(*s)) printf("push '%c'          ; Operand\n",*(s++));
        else
        {
            if (*s==';') { puts("pop  ax          ; Ergebnis ins ax-Register\n\n"); break; }
                        /* letztes Ergebnis vom Stapel und fertig... */
            puts("pop  bx\npop  ax          ; Operanden vom Stack");
            switch(*(s++))
            {
                case '=':printf("mov  word ptr [ax],bx ; Zuweisung\n"); break;
                case '+':puts("add  ax,bx          ; +"); break;
                case '-':puts("sub  ax,bx          ; -"); break;
                case '*':puts("mul  bx          ; *"); break;
                case '/':puts("clc\ndiv bx          ; /"); break;
                case '^':puts("exp  bx"); break;
            }
            puts("push ax          ; Ergebnis auf den Stack");
        }
    }
}

void main()
{
    char infix[256],postfix[256],syntaxfehler;

    puts("Bitte geben Sie die Funktion in der Infixform an:\n(Beispiel: x=a+b-c*(d+e)^7;) Bitte mit Semikolon terminieren.");
    gets(infix);
    strcpy(postfix,upn(infix));

    syntaxfehler=syntax_not_ok(postfix);
    if (syntaxfehler)
    {
        if (syntaxfehler==2) printf("Semikolon fehlt. ");
        error("Syntax nicht OK");
    }
    else
        puts("Syntax scheint OK...");

    puts("\nDie Postfixform lautet:");
    printf("%s = %s\n",infix,postfix);
    puts("\nUnd x86-,hnliche Assemblerbefehle w,,ren:");
    assemble(postfix);
}SUB

```