

```

/*****
/*
/*   LR-Testparser fuer eine kleine Grammatik
/*   (C) by Daniel Becker
/*   Letzte Aenderung 21.5.99
/*
*****/
#include <stdlib.h>
#include <string.h>

/* Grammatik:
   E'=E
   E =E+T | T
   T =T*F | F
   F =(E) | a*/

struct a
{
    char sr,i[20];

} action[11][6];

char gototab[11][3];
char translatnon[]="+*()a; ",translatter[]="ETF";
char stack[256],wort[80];
int s;

/* Stack Operationen *****/
/* auch mit #definern moeglich *****/

/*#define push(c) ((s>255)?exit(3):stack[s++]=c) */
void push(char c)
{
    if (s>255) exit(3);
    stack[s++]=c;
}

/*#define pop() ((!s)?exit(3):stack[--s]) */
char pop()
{
    if (!s) exit(3);
    return(stack[--s]);
}

/*#define toposofstack() ((!s)?exit(3):stack[s-1]) */
char toposofstack()
{
    if (!s) exit(3);
    return(stack[s-1]);
}

void initstack()
{
    s=0;
    push(0);
}

/***** Stack ende! *****/

int getindexnon(char terminal)
{
    char *s;

    s=strchr(translatnon,terminal);
    return(s-translatnon);
}

int getindexter(char terminal)
{
    char *s;

    s=strchr(translatter,terminal);
    return(s-translatter);
}

```

```

void addtoaction(int zustand, char terminal, char shrd, char *ns)
{
    int x;

    x=getindexnon(terminal);
    action[zustand][x].sr=shrd;
    if (shrd=='s') *action[zustand][x].i=atoi(ns);
    else strcpy(action[zustand][x].i,ns);
}

void addtogoto(int zustand, int terminal, char ns)
{
    int x;

    x=getindexter(terminal);
    gototab[zustand][x]=ns;
}

int lrparse(char *w)
{
    char *ip,i,a;
    int s,shlp;

    ip=w;
    initstack();
    while (1)
    {
        a=getindexnon(*ip);
        s=topofstack();
        if (action[s][a].sr=='s') /* SHIFT-FALL */
        {
            push(*ip); push(*action[s][a].i); ip++;
        }
        else
        if (action[s][a].sr=='r') /* REDUCE-FALL */
        {
            for (i=2*(strlen(action[s][a].i)-2); i>0; i--) pop();
            shlp=topofstack();
            push(*action[s][a].i);
            push(gototab[shlp][getindexter(*action[s][a].i)]);
            printf("%s\n",action[s][a].i);
        }
        else
        if (action[s][a].sr=='a') /* Accept */
            return(1);
        else
            return(0);
    }
}

void buildtabelle()
{
    addtoaction(0,'(','s',"4");
    addtoaction(0,'a','s',"5");

    addtoaction(1,'+','s',"6");
    addtoaction(1,';','a',"");

    addtoaction(2,'+','r',"E=T");
    addtoaction(2,'*','s',"7");
    addtoaction(2,')','r',"E=T");
    addtoaction(2,';','r',"E=T");

    addtoaction(3,'+','r',"T=F");
    addtoaction(3,'*','r',"T=F");
    addtoaction(3,')','r',"T=F");
    addtoaction(3,';','r',"T=F");

    addtoaction(4,'(','s',"4");
    addtoaction(4,'a','s',"5");

    addtoaction(5,'+','r',"F=a");

```

```

addtoaction(5, '*', 'r', "F=a");
addtoaction(5, ')', 'r', "F=a");
addtoaction(5, ';', 'r', "F=a");

addtoaction(6, '(', 's', "4");
addtoaction(6, 'a', 's', "5");

addtoaction(7, '(', 's', "4");
addtoaction(7, 'a', 's', "5");

addtoaction(8, '+', 's', "6");
addtoaction(8, ')', 's', "11");

addtoaction(9, '+', 'r', "E=E+T");
addtoaction(9, '*', 's', "7");
addtoaction(9, ')', 'r', "E=E+T");
addtoaction(9, ';', 'r', "E=E+T");

addtoaction(10, '+', 'r', "T=T*F");
addtoaction(10, '*', 'r', "T=T*F");
addtoaction(10, ')', 'r', "T=T*F");
addtoaction(10, ';', 'r', "T=T*F");

addtoaction(11, '+', 'r', "F=(E)");
addtoaction(11, '*', 'r', "F=(E)");
addtoaction(11, ')', 'r', "F=(E)");
addtoaction(11, ';', 'r', "F=(E)");

addtogoto(0, 'E', 1);
addtogoto(0, 'T', 2);
addtogoto(0, 'F', 3);

addtogoto(4, 'E', 8);
addtogoto(4, 'T', 2);
addtogoto(4, 'F', 3);

addtogoto(6, 'T', 9);
addtogoto(6, 'F', 3);

addtogoto(7, 'F', 10);
}

void main()
{
    buildtabelle();

    clrscr();

    printf("Bitte geben Sie ein Wort ein: ");
    gets(wort);

    if (lrparse(wort))
        puts("\nWort ist OK!");
    else
        puts("\nWort ist nicht OK!!");
}
SUB

```