**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

char prol[7][10] = {"S", "A", "A", "B", "B", "C", "C"};

char pror[7][10] = {"A", "Bb", "Cd", "aB", "@", "Cc", "@"};

char prod[7][10] = {"S->A", "A->Bb", "A->Cd", "B->aB", "B->@", "C->Cc", "C->@"};

char first[7][10] = {"abcd", "ab", "cd", "a@", "@", "c@", "@"};

char follow[7][10] = {"$", "$", "$", "a$", "b$", "c$", "d$"};

char table[5][6][10];

int numr(char c)

{

switch(c) {

case 'S': return 0;

case 'A': return 1;

case 'B': return 2;

case 'C': return 3;

case 'a': return 0;

case 'b': return 1;

case 'c': return 2;

case 'd': return 3;

case '$': return 4;

}

return (2);

}

void main()

{

int i, j, k;

for(i = 0; i < 5; i++) {

for(j = 0; j < 6; j++) {

strcpy(table[i][j], " ");

}

}

printf("\nThe following is the predictive parsing table for the following grammar:\n");

for(i = 0; i < 7; i++) {

printf("%s\n", prod[i]);

}

printf("\nPredictive parsing table is\n");

fflush(stdin);

for(i = 0; i < 7; i++) {

k = strlen(first[i]);

for(j = 0; j < 10; j++) {

if(first[i][j] != '@') {

strcpy(table[numr(prol[i][0]) + 1][numr(first[i][j]) + 1], prod[i]);

}

}

}

for(i = 0; i < 7; i++) {

if(strlen(pror[i]) == 1) {

if(pror[i][0] == '@') {

k = strlen(follow[i]);

for(j = 0; j < k; j++) {

strcpy(table[numr(prol[i][0]) + 1][numr(follow[i][j]) + 1], prod[i]);

}

}

}

}

strcpy(table[0][0], " ");

strcpy(table[0][1], "a");

strcpy(table[0][2], "b");

strcpy(table[0][3], "c");

strcpy(table[0][4], "d");

strcpy(table[0][5], "$");

strcpy(table[1][0], "S");

strcpy(table[2][0], "A");

strcpy(table[3][0], "B");

strcpy(table[4][0], "C");

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

for(i=0;i<5;i++)

for(j=0;j<6;j++){

printf("%-10s",table[i][j]); }

if(j==5) {

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

}

getch();

}

**PROGRAM:**

#include <stdio.h>

#include <string.h>

int k = 0, z = 0, i = 0, j = 0, c = 0;

char a[16], ac[20], stk[15], act[10];

void check();

int main()

{

puts("GRAMMAR is E->E+E \nE->E\*E \nE->(E) \nE->id");

puts("enter input string ");

gets(a);

c = strlen(a);

strcpy(act, "SHIFT->");

puts("stack \t input \t action");

for (k = 0, i = 0; j < c; k++, i++, j++)

{

if (a[j] == 'i' && a[j+1] == 'd')

{

stk[i] = a[j];

stk[i+1] = a[j+1];

stk[i+2] = '\0';

a[j] = ' ';

a[j+1] = ' ';

printf("\n$%s\t%s$\t%sid", stk, a, act);

check();

}

else

{

stk[i] = a[j];

stk[i+1] = '\0';

a[j] = ' ';

printf("\n$%s\t%s$\t%ssymbols", stk, a, act);

check();

}

}

}

void check()

{

strcpy(ac, "REDUCE TO E");

for (z = 0; z < c; z++)

{

if (stk[z] == 'i' && stk[z+1] == 'd')

{

stk[z] = 'E';

stk[z+1] = '\0';

printf("\n$%s\t%s$\t%s", stk, a, ac);

j++;

}

if (stk[z] == 'E' && stk[z+1] == '+' && stk[z+2] == 'E')

{

stk[z] = 'E';

stk[z+1] = '\0';

stk[z+2] = '\0';

printf("\n$%s\t%s$\t%s", stk, a, ac);

i = i - 2;

}

if (stk[z] == 'E' && stk[z+1] == '\*' && stk[z+2] == 'E')

{

stk[z] = 'E';

stk[z+1] = '\0';

stk[z+2] = '\0';

printf("\n$%s\t%s$\t%s", stk, a, ac);

i = i - 2;

}

if (stk[z] == '(' && stk[z+1] == 'E' && stk[z+2] == ')')

{

stk[z] = 'E';

stk[z+1] = '\0';

stk[z+2] = '\0';

printf("\n$%s\t%s$\t%s", stk, a, ac);

i = i - 2;

}

}

}

**PROGRAM:**

#include <conio.h>

#include <stdio.h>

char arr[18][3] = {

{'E', '+', 'F'}, {'E', '\*', 'F'}, {'E', '(', 'F'}, {'E', ')', 'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},

{'F', '+', 'F'}, {'F', '\*', 'F'}, {'F', '(', 'F'}, {'F', ')', 'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'},

{'T', '+', 'F'}, {'T', '\*', 'F'}, {'T', '(', 'F'}, {'T', ')', 'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}

};

char prod[6] = "EETTFF";

char res[6][3] = {

{'E', '+', 'T'}, {'T', '\0'}, {'T', '\*', 'F'}, {'F', '\0'}, {'(', 'E', ')'}, {'i', '\0'}

};

char stack[5][2];

int top = -1;

void install(char pro, char re) {

int i;

for (i = 0; i < 18; ++i) {

if (arr[i][0] == pro && arr[i][1] == re) {

arr[i][2] = 'T';

break;

}

}

++top;

stack[top][0] = pro;

stack[top][1] = re;

}

void main() {

int i = 0, j;

char pro, re, pri = ' ';

for (i = 0; i < 6; ++i) {

for (j = 0; j < 3 && res[i][j] != '\0'; ++j) {

if (res[i][j] == '+' || res[i][j] == '\*' || res[i][j] == '(' || res[i][j] == ')' || res[i][j] == 'i' || res[i][j] == '$') {

install(prod[i], res[i][j]);

break;

}

}

}

while (top >= 0) {

pro = stack[top][0];

re = stack[top][1];

--top;

for (i = 0; i < 6; ++i) {

if (res[i][0] == pro && res[i][0] != prod[i]) {

install(prod[i], re);

}

}

}

for (i = 0; i < 18; ++i) {

printf("\n\t");

for (j = 0; j < 3; ++j) {

printf("%c\t", arr[i][j]);

}

}

getch();

printf("\n\n");

for (i = 0; i < 18; ++i) {

if (pri != arr[i][0]) {

pri = arr[i][0];

printf("\n\t%c -> ", pri);

}

if (arr[i][2] == 'T') {

printf("%c ", arr[i][1]);

}

}

getch();

}

**PROGRAM:**

#include <string.h>

#include <conio.h>

#include <stdio.h>

int axn[][6][2] = {

{{100, 5}, {-1, -1}, {-1, -1}, {100, 4}, {-1, -1}, {-1, -1}},

{{-1, -1}, {100, 6}, {-1, -1}, {-1, -1}, {-1, -1}, {102, 102}},

{{-1, -1}, {101, 2}, {100, 7}, {-1, -1}, {101, 2}, {101, 2}},

{{-1, -1}, {101, 4}, {101, 4}, {-1, -1}, {101, 4}, {101, 4}},

{{100, 5}, {-1, -1}, {-1, -1}, {100, 4}, {-1, -1}, {-1, -1}},

{{100, 5}, {101, 6}, {101, 6}, {-1, -1}, {101, 6}, {101, 6}},

{{100, 5}, {-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}},

{{100, 5}, {-1, -1}, {-1, -1}, {100, 4}, {-1, -1}, {-1, -1}},

{{-1, -1}, {100, 6}, {-1, -1}, {-1, -1}, {100, 11}, {-1, -1}},

{{-1, -1}, {101, 1}, {100, 7}, {-1, -1}, {101, 1}, {101, 1}},

{{-1, -1}, {101, 3}, {101, 3}, {-1, -1}, {101, 3}, {101, 3}},

{{-1, -1}, {101, 5}, {101, 5}, {-1, -1}, {101, 5}, {101, 5}}

};

int gotot[12][3] = {

{1, 2, 3}, {-1, -1, -1}, {-1, -1, -1}, {-1, -1, -1},

{8, 2, 3}, {-1, -1, -1}, {-1, -1, -1}, {9, 3, -1},

{-1, -1, -1}, {10, -1, -1}, {-1, -1, -1}, {-1, -1, -1}

};

int a[10];

char b[10];

int top = -1, btop = -1, i;

void push(int k) {

if (top < 9)

a[++top] = k;

}

void pushb(char k) {

if (btop < 9)

b[++btop] = k;

}

char TOS() {

return a[top];

}

void pop() {

if (top >= 0)

top--;

}

void popb() {

if (btop >= 0)

b[btop--] = '\0';

}

void display() {

for (i = 0; i <= top; i++)

printf("%d%c", a[i], b[i]);

}

void display1(char p[],int m) {

int l;

printf("\t\t");

for(l=m;p[l]!='\0';l++) {

printf("%c",p[l]);

printf("\n"); }

}

void error() {

printf("\n\nSyntax Error");

}

void reduce(int p)

{

int len,k,ad;

char src,\*dest;

switch(p)

{

case 1:

dest="E+T";

src='E';

break;

case 2:

dest="T";

src='E';

break;

case 3:

dest="T\*F";

src='T';

break;

case 4:

dest="F";

src='T';

break;

case 5:

dest="(E)";

src='F';

break;

case 6:

dest="i";

src='F';

break;

default:

dest="\0";

src='\0';

break;

}

for(k=0;k<strlen(dest);k++)

{

pop();

popb();

}

pushb(src);

switch(src)

{

case 'E':

ad=0;

break;

case 'T':

ad=1;

break;

case 'F':

ad=2;

break;

default:

ad=-1;

break;

}

push(gotot[TOS()][ad]);

}

int main()

{

int j,st,ic;

char ip[20]="\0",an;

printf("Enter any String :- ");

gets(ip);

push(0);

display();

printf("\t%s\n",ip);

for(j=0;ip[j]!='\0';) {

st=TOS();

an=ip[j];

if(an>='a'&an<='z') {

ic=0;

} else if(an=='+') {

ic=1;

} else if(an=='\*') {

ic=2;

} else if(an=='(') {

ic=3;

} else if(an==')') {

ic=4;

} else if(an=='$') {

ic=5;

} else {

error();

break;

}

if(axn[st][ic][0]==100) {

pushb(an);

push(axn[st][ic][1]);

display();

j++;

display1(ip,j);

}

if(axn[st][ic][0]==101) {

reduce(axn[st][ic][1]);

display();

display1(ip,j);

}

if(axn[st][ic][1]==102) {

printf("Given String is Accepted");

break;

}

}

getch();

return 0;

}

**PROGRAM:**

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x) {

stack[++top] = x;

}

int pop() {

return stack[top--];

}

int main() {

char exp[20];

char \*e;

int n1, n2, n3, num;

printf("Enter the expression :: ");

scanf("%s", exp);

e = exp;

while (\*e != '\0') {

if (isdigit(\*e)) {

num = \*e - 48;

push(num);

}

else {

n1 = pop();

n2 = pop();

switch (\*e) {

case '+':{

n3 = n1 + n2;

break;

}

case '-':{

n3 = n2 - n1;

break;

}

case '\*':{

n3 = n1 \* n2;

break;

}

case '/':{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

}

printf("\nThe result of expression %s = %d\n\n", exp, pop());

return 0;

}