Infix to Postfix

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node {

    char data;

    struct node \* next;

};

#define MAX 100

char stack[MAX];

int top = -1;

//function prototypes

void push(char item);

char pop();

void peek();

int isEmpty();

int isFull();

void display();

int instackPriority(char c){

  int k;

  switch(c){

    case '(':

      k = 0;

      break;

    case '+':

    case '-':

      k = 1;

      break;

    case '\*':

    case '/':

    case '%':

      k = 2;

      break;

    case '^':

      k = 3;

      break;

  }

  return k;

}

int incomingPriority(char c){

  switch(c){

    case '(':return 0;

    case '+':

    case '-':return 1;

    case '\*':

    case '/':

    case '%':return 2;

    case '^':return 4;

  }

}

int isWhiteSpace(char c ){

  if (c==' ' || c=='\t'||c=='\n')

    return 1;

  return 0;

}

void push(char item){

  //check for overflow

  if (isFull()){

    printf("Error: Stack overflow\n");

    return;

  }

  //increment top, add item at top

  stack[++top] = item;

}

char pop(){

  //check for stack underflow

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return 0;

  }

  return stack[top--];

}

void peek(){

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return;

  }

  printf("Item on top of stack is %d\n", stack[top]);

}

int isEmpty(){

if (top == -1)

  return 1;

return 0;

}

int isFull(){

  if (top == MAX-1)

    return 1;

  return 0;

}

void display(){

  //as long as it is not empty display

  if (isEmpty()){

    printf("Empty stack\n");

    return;

  }

  int i;

  //printf("Stack contents:\n ");

  for (i = top; i >= 0; i--){

    printf("%c  ",stack[i]);

  }

  printf("\n");

}

void infixToPostfix(char \* infixExp){

  int i,p = 0;

  char next, symbol, postfix[100];

  for (i = 0;i < strlen(infixExp);i++){

    symbol = infixExp[i];

    if (!isWhiteSpace(symbol)){

      switch(symbol){

        case '(':

          push('(');

          break;

        case ')':

          while ((next=pop(stack))!='('){

            postfix[p++] = next;

          }

          break;

        case '+':

        case '-':

        case '\*':

        case '/':

        case '^':

          int i, j;

          i = instackPriority(stack[top]);

          j = incomingPriority(symbol);

          while

((top!=1)&&(instackPriority(stack[top]) >= incomingPriority(symbol)))

            postfix[p++] = pop();

          push(symbol);

          break;

        default:

          postfix[p++] = symbol;

          break;

      }

    }

  }

  while (top!= -1){

    postfix[p++] = pop();

  }

  postfix[p] = '\0';

  printf("Final expression is %s\n",postfix);

}

int main(){

  //code for evaluating postfix express

  char exp[100];

  printf("Enter expression:");

  scanf("%s",exp);

  infixToPostfix(exp);

}

**Postfix Evaluation**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define MAX 100

int stack[MAX];

int top = -1;

//function prototypes

void push(int item);

int pop();

void peek();

int isEmpty();

int isFull();

void display();

void push(int item){

  //check for overflow

  if (isFull()){

    printf("Error: Stack overflow\n");

    return;

  }

  //increment top, add item at top

  stack[++top] = item;

}

int pop(){

  //check for stack underflow

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return 0;

  }

  return stack[top--];

}

void peek(){

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return;

  }

  printf("Item on top of stack is %d\n", stack[top]);

}

int isEmpty(){

if (top == -1)

  return 1;

return 0;

}

int isFull(){

  if (top == MAX-1)

    return 1;

  return 0;

}

void display(){

  //as long as it is not empty display

  if (isEmpty()){

    printf("Empty stack\n");

    return;

  }

  int i;

  //printf("Stack contents:\n ");

  for (i = top; i >= 0; i--){

    printf("%c  ",stack[i]);

  }

  printf("\n");

}

long int pfEval(char exp[]){

  long int a,b,temp,result;

  int i;

  //use the same stack for this

  for (i = 0;i<strlen(exp);i++){

    if (exp[i]<='9' && exp[i]>='0')

      push(exp[i]-'0');

    else {

      a = pop();

      b = pop();

      switch(exp[i]){

        case '+':

          temp = b+a;

          break;

        case '-':

          temp = b-a;

          break;

        case '\*':

          temp = b\*a;

          break;

        case '/':

          temp = b/a;

          break;

        case '%':

          temp = b%a;

          break;

        case '^':

          temp = pow(b,a);

          break;

      }

      push(temp);

    }

  }

  result = pop();

  return result;

}

int main(){

  //code for evaluating postfix express

  char exp[100];

  printf("Enter expression:");

  scanf("%s",exp);

  printf("Result is %ld\n",pfEval(exp));

}

**Infix to prefix**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node {

    char data;

    struct node \* next;

};

#define MAX 100

char stack[MAX];

int top = -1;

//function prototypes

void push(char item);

char pop();

void peek();

int isEmpty();

int isFull();

void display();

int instackPriority(char c){

  int k;

  switch(c){

    case ')':

      k = 0;

      break;

    case '+':

    case '-':

      k = 1;

      break;

    case '\*':

    case '/':

    case '%':

      k = 2;

      break;

    case '^':

      k = 4;

      break;

  }

  return k;

}

int incomingPriority(char c){

  switch(c){

    case ')':return 0;

    case '+':

    case '-':return 1;

    case '\*':

    case '/':

    case '%':return 2;

    case '^':return 3;

  }

}

int isWhiteSpace(char c ){

  if (c==' ' || c=='\t'||c=='\n')

    return 1;

  return 0;

}

void push(char item){

  //check for overflow

  if (isFull()){

    printf("Error: Stack overflow\n");

    return;

  }

  //increment top, add item at top

  stack[++top] = item;

}

char pop(){

  //check for stack underflow

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return 0;

  }

  return stack[top--];

}

void peek(){

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return;

  }

  printf("Item on top of stack is %d\n", stack[top]);

}

int isEmpty(){

if (top == -1)

  return 1;

return 0;

}

int isFull(){

  if (top == MAX-1)

    return 1;

  return 0;

}

void display(){

  //as long as it is not empty display

  if (isEmpty()){

    printf("Empty stack\n");

    return;

  }

  int i;

  //printf("Stack contents:\n ");

  for (i = top; i >= 0; i--){

    printf("%c  ",stack[i]);

  }

  printf("\n");

}

void infixToprefix(char \* infixExp){

  int i,p = 0;

  char next, symbol, prefix[100];

  for (i = strlen(infixExp)-1;i >= 0;i--){

    symbol = infixExp[i];

    if (!isWhiteSpace(symbol)){

      switch(symbol){

        case ')':

          push(')');

          break;

        case '(':

          while ((next=pop(stack))!=')'){

            prefix[p++] = next;

          }

          break;

        case '+':

        case '-':

        case '\*':

        case '/':

        case '^':

          while ((top!=-1)&&(instackPriority(stack[top]) > incomingPriority(symbol)))

            prefix[p++] = pop();

          push(symbol);

          break;

        default:

          prefix[p++] = symbol;

          break;

      }

    }

  }

  while (top!= -1){

    prefix[p++] = pop();

  }

  prefix[p] = '\0';

  printf("Final expression is %s\n",strrev(prefix));

}

int main(){

  //code for evaluating prefix // expression

  char exp[100];

  printf("Enter expression:");

  scanf("%s",exp);

  infixToprefix(exp);

}

**Prefix Evaluation**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define MAX 100

int stack[MAX];

int top = -1;

//function prototypes

void push(int item);

int pop();

void peek();

int isEmpty();

int isFull();

void display();

void push(int item){

  //check for overflow

  if (isFull()){

    printf("Error: Stack overflow\n");

    return;

  }

  //increment top, add item at top

  stack[++top] = item;

}

int pop(){

  //check for stack underflow

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return 0;

  }

  return stack[top--];

}

void peek(){

  if (isEmpty()){

    printf("Error:Stack underflow\n");

    return;

  }

  printf("Item on top of stack is %d\n", stack[top]);

}

int isEmpty(){

if (top == -1)

  return 1;

return 0;

}

int isFull(){

  if (top == MAX-1)

    return 1;

  return 0;

}

void display(){

  //as long as it is not empty display

  if (isEmpty()){

    printf("Empty stack\n");

    return;

  }

  int i;

  //printf("Stack contents:\n ");

  for (i = top; i >= 0; i--){

    printf("%c  ",stack[i]);

  }

  printf("\n");

}

long int prfEval(char exp[]){

  long int a,b,temp,result;

  int i;

  char \* exp2 = (char \*)malloc(sizeof(char)\*100);

  strcpy(exp2,strrev(exp));

  for (i = 0;i<strlen(exp2);i++){

    if (exp[i]<='9' && exp[i]>='0')

      push(exp[i]-'0');

    else {

      a = pop();

      b = pop();

      switch(exp[i]){

        case '+':

          temp = a+b;

          break;

        case '-':

          temp = a-b;

          break;

        case '\*':

          temp = a\*b;

          break;

        case '/':

          temp = a/b;

          break;

        case '%':

          temp = a%b;

          break;

        case '^':

          temp = pow(a,b);

          break;

      }

      push(temp);

    }

  }

  result = pop();

  return result;

}

int main(){

  //code for evaluating postfix express

  char exp[100];

  printf("Enter expression:");

  scanf("%s",exp);

  printf("Result is %ld\n",prfEval(exp));

}