자료 구조

[프로그래밍 과제 #1] 수식 표현의 계산

제출일자 : 2017.10.11

담당교수 : 이경아 교수님

분반번호 : 01

학과 : 컴퓨터공학과

학번 : 1615051

이름 : 이영은

연락처 : 010-6729-0907

자료구조 레포트

1. 문제 기술

: 중위 표기법으로 표현 된 여러 개의 수식들을 입력을 받아서 후위 표기법으로 변환 후, 변환한 수식의 값을 계산하는 문제이다. 중위 표기법과 다르게 후위 표기법에서는 괄호가 필요 없다. 따라서 컴파일러는 후위 표기 방법을 선호하기 때문에 이러한 수식의 문제를 해결할 때 좀 더 수월하게 이용하는 프로그램을 사용하는 경우가 있다.

2. 입출력의 예

: 파일 infix1에서 첫번째 수식인 ‘2\*3/3+4-5’를 예로 들어서 설명하겠다. 먼저 ‘2\*3/3+4-5’를 입력한다. 그 다음 중위 표기식을 후위 표기식으로 바꾸는 프로그램을 이용하여 후위 표기식으로 바꿔준다. 따라서 출력은 ‘2 3 \* 3 / 4 + 5 –‘가 된다.

3. 문제풀이 방법(알고리즘)

: 먼저 main 함수에서 txt파일을 연다. 그 다음 배열 str을 초기화 한 후, 먼저 fgetc(fp)를 이용하여 주어신 수식의 개수를 표현하는 숫자를 가져와서 ‘0’를 빼서 변수 n이라는 int값으로 정의한다. 그 다음 가장 중심이 되는 for문을 사용한다. 이 for문에서는 앞에서 정의한 n까지 for문을 실행하는데, 이는 수식의 개수만큼 실행된다는 것을 의미한다. 그 다음 중심이 되는 for문을 분석해본다. 먼저 fgetc(fp)를 이용하여 수식의 제일 첫번째 문자를 받아들인다. 그리고 while문의 조건인 ‘가져온 문자가 ‘;’이 아니면 while의 반복문을 실행한다. 예를 들어 내가 받아들인 문자가 2라고 한다면 조건을 만족하기 때문에 while의 반복문을 실행한다. 먼저 배열 str[0]에 문자 2를 저장하고 postfix를 이용하여 j에 +1을 한다. 그 다음 fegtc(fp)하여 가져온 문자 한 개를 출력한 후, 그 다음 배열인 str[1]을 NULL로 저장한다. 이러한 while 반복문을 이용하여 “infix notation”을 표현한다.

그 다음 postfix notation을 표현한다. main에서는 간단하게 ‘infix\_to\_postfix(str)’로 적은 후, 다른 함수의 infix\_to\_postfix(str)을 실행한다. 먼저 int와 char로 변수를 선언 후 strlen으로 배열의 길이(수식의 길이)를 int len으로 선언한다. 그리고 StackType인 s를 타입 정의한다. init(&s)를 이용하여 현재 갖고있는 스택을 초기화 한다. 먼저 중심이 되는 for문에서 수식의 길이까지 반복이 진행된다는 것을 정의해준다. 반복문 for문에서 지정한 배열을 ch로 반복이 끝날 때 까지 받는다.

1) 만약 ch가 연산자가 나오면, 반복문 while을 사용하는데 스택에 있는 연산자의 우선순위가 더 크거나 같으면 while문은 반복하여 출력된다. 이 반복문에서는 stack에서 pop한 문자를 배열 str1에 저장하고 이것을 k++하여 배열이 끝까지 저장해준다. 그 다음 push를 이용하여 stack에 입력된 연산자를 넣는다.

2) 만약 ‘(‘가 입력되면 stack에 ‘(‘를 집어넣는다.

3) 만약 ‘)’가 입력되면 stack에 맨 위에 저장되어 있는 값을 top\_op에 저장한다. 그리고 top\_op에넣은 문자가 ‘(‘가 아니면 반복문 while을 실행하는데 이렇게 pop한 값을 배열 str1에 저장하고 printf를 이용하여 출력한다. 그 다음 top\_op에 stack에서 꺼낸 문자를 저장하고 k++을 해준다. 이렇게 stack에 저장되어있는 값에서 pop하여 나온 값이 ‘(‘이 나올 때까지 반복문을 실행하고 break를 하여 멈춘다.

4) 만약 ch가 피연산자라서 case에 부합하지 않는 경우라면, 문자 하나(숫자)를 먼저 출력하고 출력한 숫자를 배열에 저장한다.

중심이 되는 for문이 끝나면 while문이 실행되는데 이 반복문은 stack이 비어있지 않으면 반복문이 계속 실행된다. 그 다음 while문이 끝나면 배열 str1을 NULL값으로 초기화 해준다.

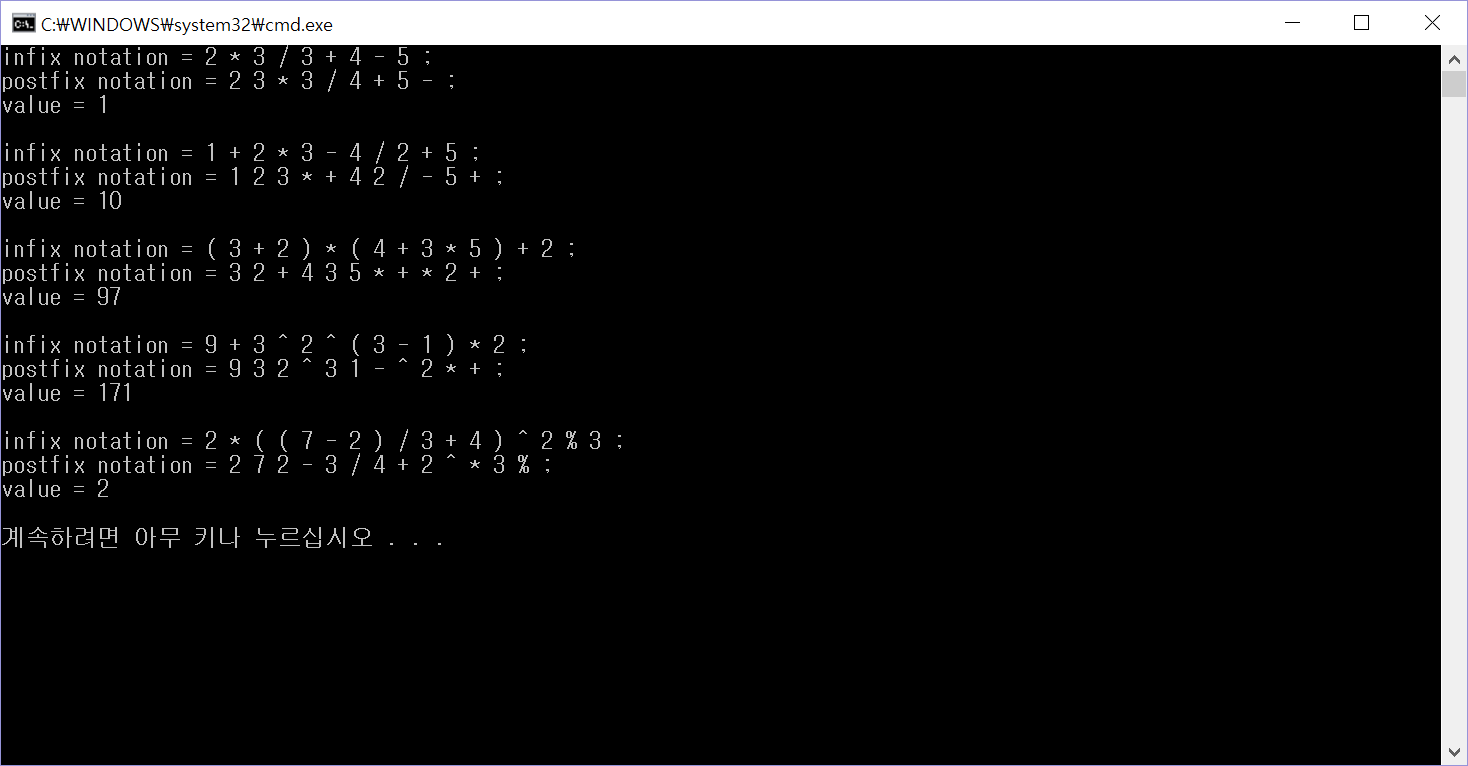
다시 main함수로 돌아와서 함수 eval(str1)을 result로 선언해준다. 함수 eval은 infix\_to\_postfix함수에서 배열에 저장한 후위 표기식 수식을 이용하여 계산해준다. 여러가지 변수를 선언 후 init(&s)를 이용하여 스택을 초기화해준다. 먼저 중심으로 for문이 이용되는데 이는 배열의 길이만큼 이용이 된다. 그 다음 만약 배열에서 입력받은 ch가 연산자가 아니라면, 문자로 선언되어있는 것을 ‘0’을 빼서 int값인 변수 value으로 저장 후 stack에 push하여 쌓아놓는다. 만약 연산자라면 stack에 쌓여있는 숫자를 pop한 후 연산자가 무엇인지 결정이 되면 계산된 값을 stack에 다시 push하여 넣는다. 이러한 조건문과 반복문이 끝나면 return값으로 stack에 쌓여있는 값을 return한다. 이렇게 return한 값을 main 함수에서 printf를 이용하여 출력해준다.

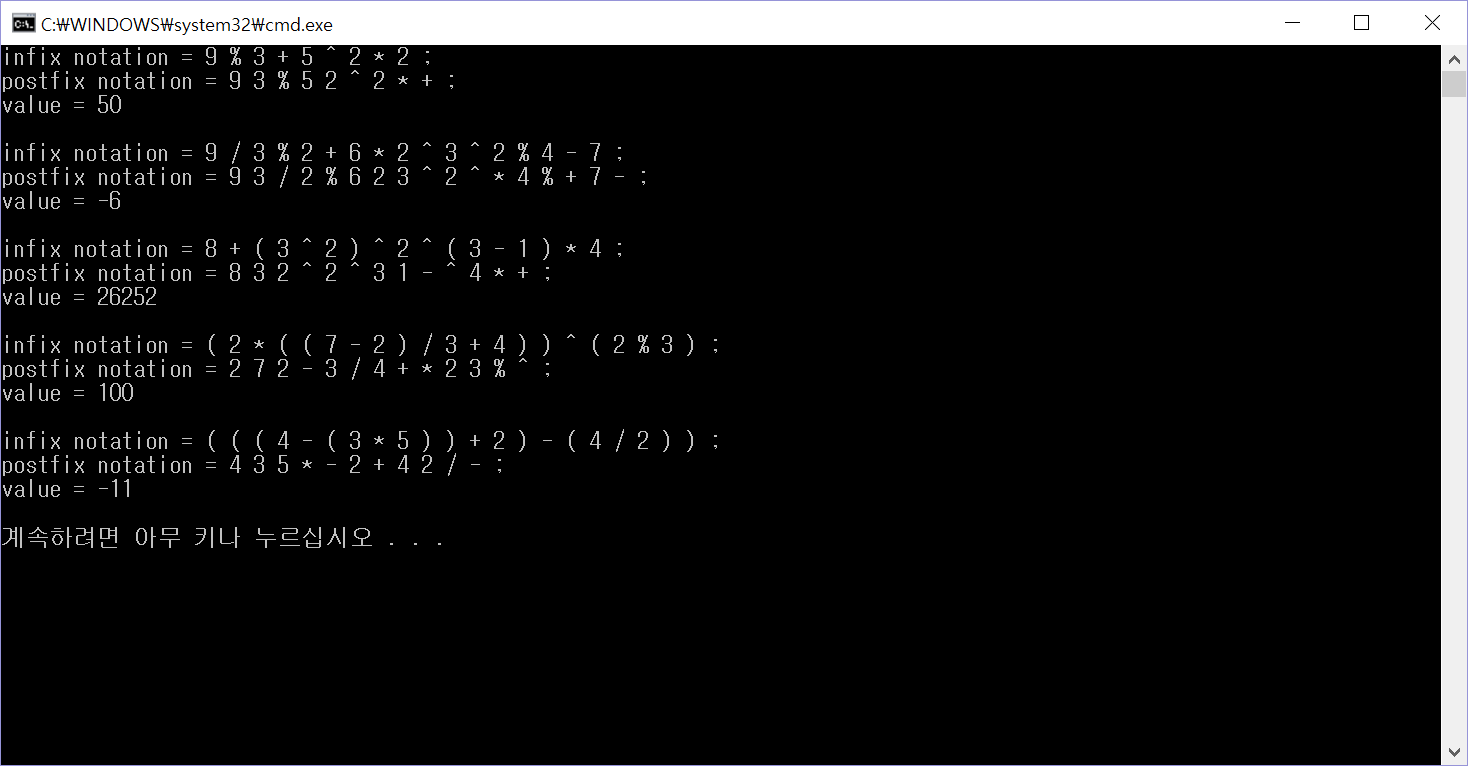
이러한 반복문과 조건문이 끝나면, fclose를 이용하여 open해놨던 파일을 닫는다.

4. 소스 프로그램

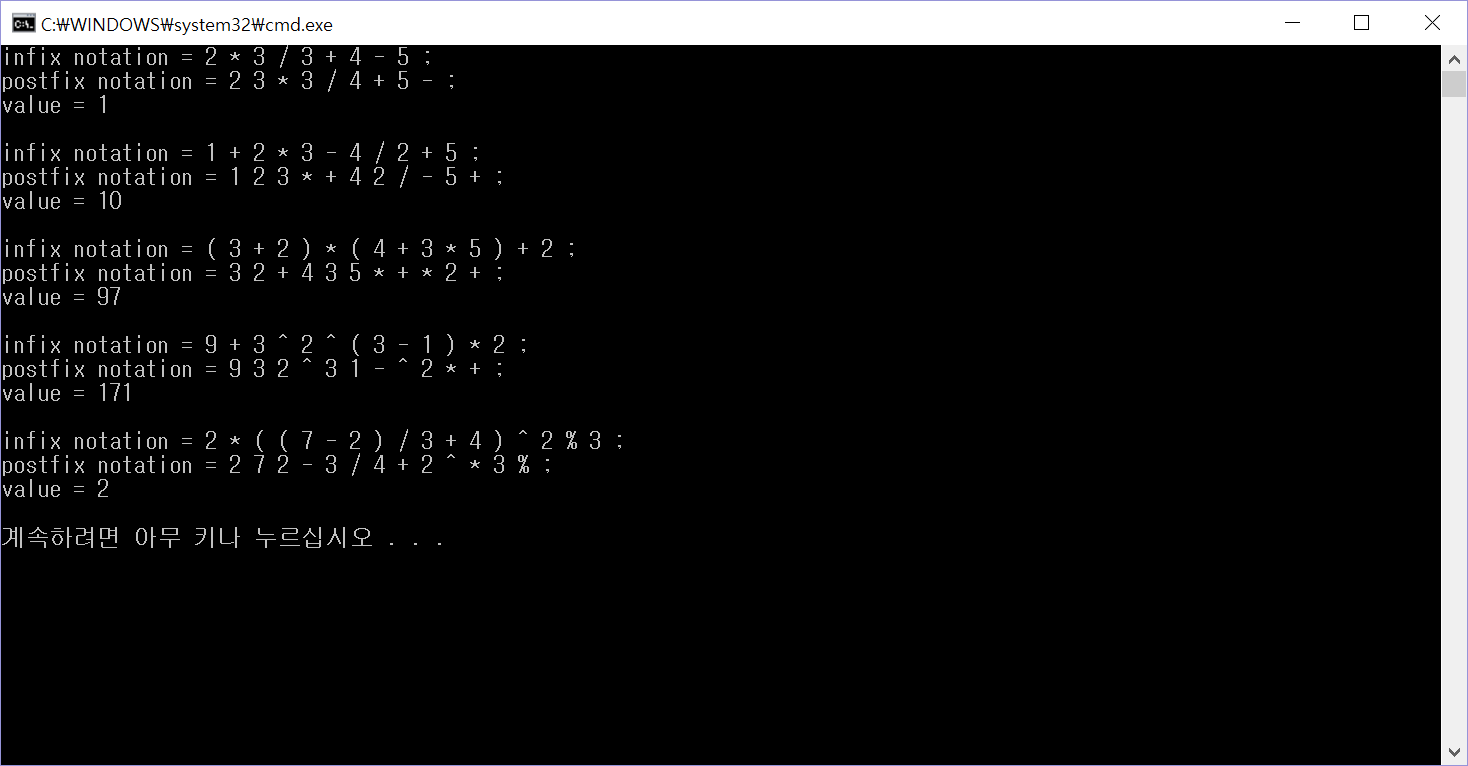
|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <math.h>  #define MAX\_STACK\_SIZE 100  typedef int element;//기존의 자료형에 int성질을 갖고있고 element라는 이름을 갖는 자료형을 만든다.  typedef struct{  element stack[MAX\_STACK\_SIZE];//element 자료형에 stack이라는 배열을 선언한다.  int top;  }StackType;//StackType이라고 타입 정의한다.  char str1[100]={0};//광역변수 배열인 str1을 초기화 시켜준다.  //스택 초기화 함수  void init(StackType \*s)  {  s->top=-1;  }  //공백 상태 검출 함수  int is\_empty(StackType \*s)  {  return(s->top==-1);  }  //포화 상태 검출 함수  int is\_full(StackType \*s){  return (s->top==(MAX\_STACK\_SIZE-1));  }  //삽입 함수  void push(StackType \*s,element item){  if(is\_full(s)){  fprintf(stderr,"스택 포화 에러\n");  return;  }else s->stack[++(s->top)]=item;  }  //삭제 함수  element pop(StackType \*s){  if(is\_empty(s)){  fprintf(stderr, "스택 공백 에러1\n");  exit (1);  } else return s->stack[(s->top)--];  }  //피크 함수  element peek(StackType \*s){  if(is\_empty(s)){  fprintf(stderr,"스택 공백 에러2\n");  exit(1);  }else return s->stack[s->top];  }  //연산자의 우선순위를 반환  int prec(char op)  {  switch (op) {  case '(': case ')': return 0;  case '+': case '-': return 1;  case '\*': case '/': case '%': return 2;  case '^': return 3;  }  return -1;  }  //후위 표기 수식 계산 함수  element eval(char exp[]){  int op1,op2,value,i=0;  int len=strlen(exp);  char ch;  StackType s;  init(&s);  for(i=0;i<len;i++){  ch=exp[i];  if(ch!='+'&&ch!='-'&&ch!='\*'&&ch!='/'&&ch!='%'&&ch!='^'){  value=ch - '0'; //char에 '0'을 빼면 int값이 됨  push(&s,value);  }else{  op2=pop(&s);  op1=pop(&s);  switch(ch){//ch가 어떤 기호이냐에 따라, 연산을 수행하고 스택에 저장  case '+': push(&s,op1+op2); break;  case '-': push(&s,op1-op2); break;  case '\*': push(&s,op1\*op2); break;  case '/': push(&s,op1/op2); break;  case '%': push(&s,op1%op2); break;  case '^': push(&s,pow(op1,op2)); break;  }  }  }  return pop(&s);  }  //중위 표기 수식을 후위 표기 수식으로 바꾸는 함수  void infix\_to\_postfix(char exp[]){  int i=0,k=0;  char ch,top\_op;  int len=strlen(exp);  StackType s; //변수 s를 타입 정의한다.  init(&s);//스택 초기화    for(i=0; i<len; i++){  ch=exp[i];//char인 ch를 배열로 받는다.  switch(ch)/\*ch의 연산자가 무엇인지에 따라 결정\*/{  case '+': case '-': case '\*': case '/': case '%': case '^':  //스택에 있는 연산자의 우선순위가 더 크거나 같으면 출력  while(!is\_empty(&s)&&(prec(ch)<=prec(peek(&s)))){  str1[k]=pop(&s);//배열 str1에 pop한 char를 저장  printf("%c ",str1[k]);//배열 str1의 문자를 출력  k++;  }  push(&s,ch);//타입 정의한 s에 문자 ch를 넣는다  break;  case '(':  push(&s,ch);//타입 정의한 s에 문자 ch를 넣는다  break;  case ')':  top\_op=pop(&s);//타입 정의한 s에서 뺀 문자 ch를 변수 top\_op에 넣는다  while(top\_op!='(')/\*pop하여 top\_op에 넣은 문자가 '('가 아니면 반복문을 실행\*/{  str1[k]=top\_op;//pop하여 저장한 top\_op 문자값을 str1 배열에 저장  printf("%c ",str1[k]);//저장한 str1 배열을 출력  top\_op=pop(&s);//pop한 stacktype 값을 top\_op에 저장  k++;  }  break;  default://만약 case에 부합하지 않는 것일때 사용(숫자일때)  printf("%c ",ch);  str1[k]=ch;//숫자를 배열 str1에 저장  k++;  break;  }  }  //stacktype s가 비어있지 않으면 반복문이 계속 실행된다.  while(!is\_empty(&s)){  str1[k]=pop(&s);//pop한 stacktype을 str1의 배열에 저장  printf("%c ",str1[k]);  k++;  }  str1[k]=NULL;//반복문을 실행 후 스택에 저장되어있는 수식을 초기화하는 역할을 한다.  }  void main(){  FILE \*fp=fopen("c:\\infix1.txt","r");//출력 스트림의 형성  int result, value;  char ch;  if(fp==NULL)/\*만약 출력하려는 파일이 지정한 위치에 존재하지 않으면\*/{  printf("이 파일은 존재하지 않습니다\n");  }  else{  int n,i,j=0,k=0;  char ch;  char str[100]={0};//배열 str을 초기화한다.  n=fgetc(fp)-'0';//지정한 파일 fp에서 fgetc하여 가져온 character에 '0'을 빼서 int값으로 만들어준다.    for(i=0;i<n;i++){  fgetc(fp);//파일 fp에서 임의로 한개의 문자를 가져온다.  printf("infix notation = ");  j=0;  while((ch=fgetc(fp))!=';')/\*가져온 문자가 ';'이 나올때 까지 반복문 사용\*/{  str[j++]=ch;//str배열에 fgetc하여 가져와서 ch에 선언한 문자 하나를 저장한다  printf("%c ",ch);//문자 ch를 출력  str[j]=NULL;//j가 +1된 str의 배열을 NULL값으로 저장한다.  }  printf(";\n");  printf("postfix notation = ");  infix\_to\_postfix(str);//함수 infix\_to\_postfix에 배열 str을 넣어서 출력한다.  printf(";\n");  result=eval(str1);//함수 eval에 배열 str1을 넣어서 출력한 int값을 result로 선언한다.  printf("value = %d\n",result);  printf("\n");  }  }    fclose(fp); );//파일을 닫는다.  return;  } |

5. 수행 결과

1) 

2) 

3)



6. 결과 분석 및 토의

: 결과인 infix notation과 postfix notation, 그리고 value값이 잘 출력 되었다. infix1.txt의 첫번째 수식인 ‘2\*3/3+4-5;’를 예로 결과를 분석해보겠다. main에서 for문을 이용하여 infix notation의 각각의 문자들을 한칸씩 띄어쓰기 하여 수식을 보기 편하게 만들었다. 그 다음 postfix notation은 후위 표기법을 이용하여 표현이 되었는데 이 또한 보기 편하게 하기 위해 띄어쓰기를 이용하여 결과값을 출력하였다. 그 다음 value값은 먼저 함수 infix\_to\_postfix에서 후위 표기법으로 바꾼 배열로 저장하여 저장한 배열 값을 eval 함수에 넣어 value 값을 출력하였다.

이번 프로그래밍 레포트 작성을 위해 코드를 만드는데 계속해서 ‘스택 공백 에러’가 출력이 되었다. 그래서 puts(str)을 이용하여 배열 str에 어떻게 후위 표기식이 저장이 되는지 확인을 해봤는데 초기화가 되는것이 아닌, stack에 지속적으로 수식들이 쌓이고 이를 처리하는 eval 함수가 오류가 난다는 것을 알았다. 분석을 해보니, 배열에서 끝에 ‘\0’이 항상 붙는데 이것을 신경쓰지 않으니 pop이 되지않고 계속해서 쌓인다는 것을 알았다. 이번 프로그래밍 과제를 통해서, 배열와 변수의 초기화의 중요성을 깨닫게 되었다. 또한 텍스트 파일을 받아서 배열에 저장하는 코드를 새롭게 알게되었다.