DATA STRUCTURES AND ALGORİTHMS

HALİL ER

16011021

STACK

Method

After reading assignment as a string, this string turn to postfix. If there are variables in postfix, they convert to integer. In the end, using postfix and values of variables for calculating.

Technic

There is a struct for the stack. Two functions were used in the program, pushing to and poping from stack. There are strings for infix, postfix and variables.

The conversion to postfix was done in a while loop. Every char in infix string comparing to symbols, then doing necessary steps. After infix string finished, all things in stack poping.

Before calculating reading variables values. The values are hiding in a integer array.

Code:

```
/*
Halil ER
16011021
*/

#include <stdio.h>
#include <stdlib.h>



typedef struct s{
	char A[100];
	int top;
}Stack;

char pop(Stack*);
void push(Stack*,char);
void printStack(Stack);
char * popAddr(Stack *X);
int main(int argc, char *argv[]) {
	Stack X;
	X.top=0;
	char infix[50];
	char postfix[50],tmp;
	char variables[10]; // variables hiding
	printf("Input has to be like c = a + ( 5 * 3 ) / 4 \nAfter writing input press enter\nAssignment:\n");
	gets(infix);
	int i=0,j=0,k=0,l;
```

```
while(infix[i]!='=')
        i++;
i+=2;
int flag;  // using for numbers and variables

while(infix[i]!='\0'){

        flag=0;
        if(infix[i]==' '){
                postfix[j]=' ';
                j++;
                i++;
                flag=1;

        }

        if(infix[i]=='('){
                push(&X,'(');
                i++;
                flag=1;

        }
        if(infix[i]==')'){
                tmp=pop(&X);
                while(tmp!='('){
                        postfix[j]=tmp;
                        j++;
                        postfix[j]=' ';
                        j++;
```

```
                tmp=pop(&X);
        }
        j--;
        flag=1;
        i++;

}

if(infix[i]=='*'){
        push(&X,'*');
        i++;
        flag=1;

}
if(infix[i]=='/'){
        push(&X,'/');
        i++;
        flag=1;

}

if(infix[i]=='+'){
        tmp=pop(&X);
        while(tmp=='*' || tmp=='/'){
                if(tmp=='*' || tmp=='/'){
                        postfix[j]=tmp;
                        j++;
                        postfix[j]=' ';
                        j++;
```

```
                }else
                        push(&X,tmp);
                tmp=pop(&X);
        }
        push(&X,tmp);

        push(&X,'+');
        i++;
        flag=1;

}
if(infix[i]=='-'){
        tmp=pop(&X);
        while(tmp=='*' || tmp=='/'){
                if(tmp=='*' || tmp=='/'){
                        postfix[j]=tmp;
                        j++;
                        postfix[j]=' ';
                        j++;
                }else
                        push(&X,tmp);
                tmp=pop(&X);


        }
        push(&X,tmp);
        i++;
        push(&X,'-');
        flag=1;
```

```
        }
        if(flag==0){
                postfix[j]=infix[i];
                if(!isdigit(infix[i])){
                        while(l<10){
                                if(infix[i]==variables[l])
                                l=11;
                                else
                                l++;
                        }
                        if(l!=11)
                                variables[k++]=infix[i];
                }
                j++;
                i++;


        }
        printf("Postfix:");
        puts(postfix);
        printf("Stack:");
        printStack(X);
}
postfix[j]=' ';
j++;
while(X.top!=0){
```

```
            postfix[j]=pop(&X);
            j++;
            postfix[j]=' ';
            j++;
            printf("Postfix:");
            puts(postfix);
            printf("Stack:");
            printStack(X);
}
printf("\nPostfix:\n");
puts(postfix);
printf("\n\n");
puts(variables);
int vartoint[10]; // using for use variables as a integer

for(i=0;i<k;i++){ // reading variables values
            printf("\n%c:",variables[i]);
            scanf("%d",&vartoint[i]);
}
i=0;
j=0;
l=0;
char tmp_cti[2];
int tmp1,tmp2;

/*while(i < 20 && postfix[i]!='\0'){
            flag=0;
            while(postfix[i]==' ')
                        i++;
```

```
if(isdigit(postfix[i])){
	if(isdigit(postfix[i+1])){
		tmp_cti[0]=postfix[i];
		tmp_cti[1]=postfix[i+1];

		push(&X,atoi(tmp_cti));
		i+=2;
	}else{
		tmp1=atoi(&postfix[i]);
		//printf("%d\n",tmp1);

		push(&X,postfix[i]);
		printStack(X);
		i++;
	}
	flag=1;
}
if(postfix[i]=='+'){
	char * ja=popAddr(&X);
	char *la=popAddr(&X);
	printf("toplama %c  %c\n",j , l);
	tmp1=atoi(la);
	tmp2=atoi(ja);
	printf("%d %d\n",tmp1,tmp2);
	tmp2=tmp1+tmp2;
	system("pause");

	push(&X, tmp2);
```

```
            printf("adfadf");
            i++;
            flag=1;
    }

    if(postfix[i]=='-'){
            tmp1=pop(&X);
            tmp2=pop(&X);
            tmp2-=tmp1;
            push(&X,tmp2);
            i++;
            flag=1;
    }
    if(postfix[i]=='*'){
            tmp1=pop(&X);
            tmp2=pop(&X);
            tmp2*=tmp1;
            push(&X,tmp2);
            i++;
            flag=1;
    }

    if(postfix[i]=='/'){
            tmp1=pop(&X);
            tmp2=pop(&X);
            tmp2+=tmp1;
            push(&X,tmp2);
            i++;
            flag=1;
```

```
                }
                if(flag==0){
                        while(postfix[i]!=variables[l])
                                l++;
                        push(&X,vartoint[l]);
                        l=0;
                        i++;
                }

        }
        j=pop(&X);
        tmp1=atoi(&j);
        printf("%d",tmp1);*/
        // couldn't solve problems

        return 0;
}
char pop(Stack *X){
        char value=X->top[X->A-1];
        X->top--;
        return value;
}

char * popAddr(Stack *X)
{
        char * value= &X->top[X->A-1];
        X->top--;
        return value;
}
```

```
void push(Stack *X,char variable){
	X->A[X->top]=variable;
	X->top++;
}
void printStack(Stack X){
	int i;
	for(i=0;i<X.top;i++)
		printf("%c ",X.A[i]);
	printf("\n");
}
```