Veri Yapıları ve Algoritmalar Ödev 5

Ödev Konusu:

Stack yapısı kullanılarak infix verilen ifadenin önce postfixe çevirilip sonrasında sonucunun hesaplanarak çalışma esnasında sonuçların bir değişkende tutularak temel hesaplamalar yapılabilen bir ortam tasarlanması

Numan Kemal Kazancı
14011092

Yöntem

Öncelikle infix verilen ifade ifade postfix ifadeye çevirilir.

Bunun için yapılması gereken adımlar aşağıdaki gibidir:

1. Girilen ifadenin '=' den sonraki kısmının alınması
2. İfadenin sonuna gelene kadar her karakterin kontrol edilmesi
   a. Eğer karakter boşluksa geçilir.
   b. Eğer karakter sol parantez ise yığına push edilir.

   c. Eğer karakter sağ parantez ise sol parantez çıkana kadar yığından pop işlemi yapılır. Alınan işlem işareti postfix ifadeye eklenir. Sol parantez görüldüğünde pop işlemine son verilir. Parantezler postfix'e eklenmez.

   d. Eğer karakter sayı veya değişken ismi ise postfix ifadeye eklenir.

   e. Eğer karakter işlem işareti ise

      i. Yığının en üstünde sol parantez varsa veya en üstteki işaretin önceliği gelen işaretten düşük ise işlem işareti yığına push edilir.

      ii. Gelen işaretin önceliği en üstteki işaretin önceliğinden daha düşük ise yığındaki gelen işaretten yüksek öncelikli bütün işaretler için pop işlemi yapılır.

      iii. Stackten pop edilen işaretler postfix ifadeye eklenir.

      iv. İşlem işareti yığına push edilir.

   f. Ifadeler bittiğinde yığındaki işaretler sıra ile pop edilerek postfix ifadeye eklenir.
3. Posfix ifadenin döndürülmesi

Kullanıcının girdiği ifade postfixe çevirildikten sonra sonucun çözülerek sonucun hesaplanması gerekmektedir. Bu işlemi yine yığın veri yapısı kullanarak çözebiliriz. Bunun için aşağıdaki adımları takip ederiz:

1. Postfix ifadenin sonuna gelene kadar soldan saga doğru değerlendirilir.

2. Eğer karakter boşluksa geçilir.

3. Eğer karakter sayı ise yığına push edilir.

4. Eğer karakter değişken ise değişkenin değeri yığına push edilir.

5. Eğer karakter işlem işareti ise yığından iki değer pop edilir ve ikinci çekilen değer önce olmak üzere işlem gerçekleştirilir ve sonuç yığına push edilir.

Uygulama

Yığın işlemlerine ait tanımlamalar yapıldı ve Pop, push, peek gibi fonksiyonlar yazılarak bir yığın yapısı kuruldu.

Çalışma sırasında girilen değişkenlerin saklanması için aşağıdaki gibi bir yapı kuruldu.

```
//a struct for one variable
typedef struct Var {
	char name;
	int value;
}Var;

//a struct for keeping multiple variables (List)
typedef struct Variables {
	Var *arr;
	int count;
	int maxSize;
}Variables;
```

Var yapısı bir değişkenin bilgisini tutar, Variables yapısı ise birden fazla değişken listesi gibi düşünebiliriz.

Kullanıcı bir ifade girdiğinde eşitliğin solundaki değişkenin ismi değişkenler listesinde aranır, eğer bulunamazsa yeni bir değişken oluşturulur ve listeye eklenir.

Kullanıcının kolay bir şekilde programı kullanabilmesi için belli komutlar tanımlanmıştır. Bunlar :

```
-help :      yardım ekranını yazdırır.
-clear :     ekranı temizler.
-clearall :  Tüm değişkenleri siler.
-printall :  Tüm değişkenleri ekrana yazdırır.
-readfile :  "expressions.txt" dosyasından ifadeler okur.
-exit :      programı sonlandırır.
```

Sonuç

Sonuç olarak verilen infix ifadenin çözümlenerek sonucuna ulaşılabilmetedir.
Algoritmanın karmaşıklığı O(N) dir.

```
-----------------------------------------------------------------------
-help                           :help
-clear                          :clear screen
-clearall                       :clear all variables
-printall                       :print all variables to the console
-readfile                       :read expressions from "expressions.txt" file
-exit                           :exit
       Example Input := a = 4 + 3 ;
-----------------------------------------------------------------------
-readfile
Infix => a :=  3 ;

Postfix => a := 3

Result => a := 3

Infix => b :=  2 ;

Postfix => b := 2

Result => b := 2

Infix => c :=  a * ( b + 4 ) ;

Postfix => c := a b 4 + *

Result => c := 18

Infix => d :=  a * a + c - b*2;
Postfix => d := a a * c b 2 * - +

Result => d := 23

File read successful.
```