CMPE-231 DATA STRUCTURES

Laboratory Work # 3

3

Duration – 100 minutes Spring 2009

This laboratory work covers Binary Search using array of structure, and Stack Data Structure(-s)

✧ Experiment 1 ***(PreLab task – to be prepared by Assistant)***

Using the below program which is reading a data file(student.txt file) from the disk into array of structure which has 10 elements .

Assume that data is sorted based on student number.

Complete the below program for ***binary search*** operation.

- Read a student from the monitor and find and list his name , grade and the position in the array of structure .

```
#include <stdio.h>
struct studinfo
{
        int stnr;
        char name[10];
        int grade;
};

void main()
{
struct studinfo s[10];
int cnt;


FILE *stud_file;

cnt=0 ;

stud_file = fopen("C:\\cmpe231\\2009Spring\\student.txt" , "r");

while ( !feof(stud_file))
{
  fscanf(stud_file ,"%d %s %d" ,&s[cnt].stnr,&s[cnt].name,&s[cnt].grade);
  cnt++;
 }
fclose(stud_file);
}
```

Assume that file has the following data in it(Prepare 10 record of data sorted by student number)

```
123   Ahmet      88
456   Cemal      99
633   Sevgi      75
776   Meral     66
792   Kemal     88
..
..
8879  Meral     81
```

Experiment 2 ***(PreLab task – to be prepared in advance).***
Run and discuss about the output of the program

A vowel means *push* and a consonant means *do nothing, and an asterix means pop* in the sequence.

E A S * Y * Q U E * * S T I O * N *

Give the sequence of values returned by the *pop* operations (see C code below).

Solution: The appearance of the first asterisk symbol (input sequence is scanned from left to right) means that the preceding it character **A** is popped. Afterwards, second appearing asterisk forces **E** to be popped. Following two characters **U** and **E** are pushed onto the stack (**E** is currently on the top), while two sequential asterisks result in repeating pop operation two times, i.e. **E**, and **U** are popped. Similar observations lead us to the solution: the sequence of values returned by pop operations is

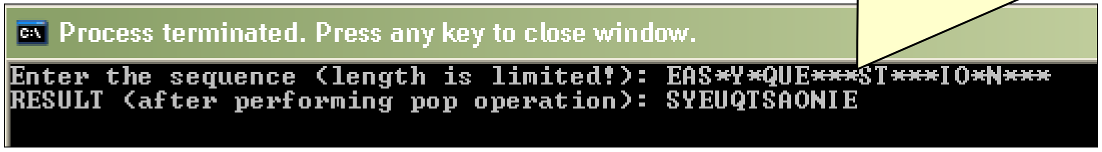
A E E U O I.

Give a way to insert asterisks in the sequence **F E N E R B A H C E** so that the sequence of values returned by the pop operations is

a) **E A E E** b) **E E A E**

```
/*************************************************/
/*  Stack's Push/Pop operations -                */
/*  1D-array implementation of stack structure   */
/*************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>

typedef struct
{
   char home[35];
   int top;
} My_stack;

void push(My_stack *, char); /* prototypes (declarations) of functions in use */
void pop(My_stack *);        /* they implement standard operations on stack */
int is_empty(My_stack *);

void main(void)
{
        My_stack st;
        char input[50];  /* input sequence for processing */
        int i = 0;
        st.top = -1;
/* previous assignment indicates that stack is empty from the beginning */
        printf("Enter the sequence (length is limited!): ");
        gets(input);
        printf("RESULT (after performing pop operation): ");

        while(input[i] != '\0')  /* while end of line is not reached */
        {
                if(isalpha(input[i]))
                        push(& st, input[i]);  /* push character onto the stack */
                else
                        if(input[i] = = '*')
                                pop(&st);
                        else
                        {
                           printf("ERROR: Wrong input symbol – program"
                            " terminates!\n");
                             exit(1);
                        }
                i++;
        }
}
void push(My_stack * s, char c)   // push (insert) operation
{  // assume there is enough space for pushing next element!
        s -> top ++;
        s -> home[s -> top] = c;
}
void pop(My_stack * s)   // pop (remove) operation
{
        if(is_empty(s))
        {
                printf("ERROR: Nothing to pop - program terminates\n");
                exit(1);
        }
        printf("%c", s -> home[s -> top - -]);
}
int is_empty(My_stack * s)   // checking whether stack is empty or not
{
        return(s -> top < 0 ? 1 : 0);
}
```

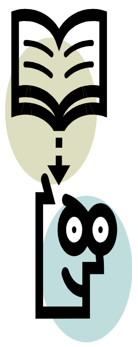
Result of a sample run (CodeWarrior C/C++ 9.2, OS Windows XP)



```
Process terminated. Press any key to close window.
Enter the sequence (length is limited!): EAS*Y*QUE***ST***IO*N***
RESULT (after performing pop operation): SYEUQTSAONIE
```

Note: Remember that «a stack is a list in which nodes can be added only at one end. Nodes can be accessed or removed only from that same end. When a new node is added it hides the previous end node, so in order to access that previous node the new node must first be removed. That is, the list works in a **Last-In First-Out** (LIFO) manner. The most recently added node is the first available for use, while the first node added is the last one available for use.

… By implementing the stack as an array you have a fixed limit on its size. And while you could obtain the array at runtime using `malloc()`, it still might be useful to have a more general solution.



«There are two kinds of facilities <*in C*> for handling characters: classification and conversion. Every character classification facility has a name beginning with **is** and returns a value of type **int** that is nonzero (true) if the argument is in the specified class and zero (false) if not. <*For example*>, the **isalnum** function tests whether its argument is an alphanumeric character – that is, one of the following in the C locale:

0 1 2 … 9, A B C … Y Z, a b c … y z.

This function is by definition equivalent to **isalpha(c)**||**isdigit(c)** (the **isalpha** function tests whether **c** is an alphabetic (upper- or lowercase) character. … In traditional C, these functions take an argument of type **char**, but they return **int**».

Source: S.P.Harbison, G.L.Steele. *C: A Reference Manual*, 5th edition, Pearson Education, 2002, 534 p.

✧ Experiment 3 *(PreLab task – to be prepared in advance)*

Write a COMPLETE C program that determines if an input character string is of the form

x * y

where **x** is a string consisting of the letters **'A'** and **'B'**, and where **y** is the reverse of **x** (that is, if **x** = **"ABABBA"**, **y** must equal **"ABBABA"**). At each point you may read only the next character of the string. **If the users enters a character other then 'A' 'B' or *, you should give error message and exit.**

For the sake of simplicity, assume that the input sequence of characters (user's input) is stored in 1D character type array (in other words, array serves as a home for characters).

HINT: The algorithm to perform required operation can be written as follows

```
while(char != *)
{
push(stack, char)
char = next_char
}
while(char != end_of_processed_string)
{if(char != pop(stack))  {
                                error
                                break
                        }
                        char = next_char;
 }
```

Result of display message will be "The string is valid" or "The string is invalid".

Example :

Input string :	**abbccba*abccbba**	message will be ***The string is valid***
	a**aabbcb*bcbaab**	message will be ***The string is invalid***
	a**adbxcy*ycxbdaa**	message will be ***Wrong character!!!***