Rajshahi University of Engineering & Technology

Department of Computer Science of Engineering

**Experiment No**: 05 & 06

**Name of Experiment**: Two Way Linked Lists & Stack

|  |
| --- |
| Submitted to:  Rizoan Toufiq  Assistant Professor  Department of Computer Science & Engineering  Rajshahi University of Engineering & Technology |

|  |
| --- |
| Submitted by:  Name: Md. Ariful Islam  Roll No.: 1803046  Group: 2nd Thirty  Date of Exp.: 14-10-2019  Date of sub. : 02-11-2019  Series: 18 Series |

Machine configuration:

ASUS X510UF

CORE I5 8th Gen Processor

Up To 3.4 GHz

8 GB RAM

OS WIN 10

INDEX

|  |  |  |
| --- | --- | --- |
| Preblem No. | Name of Problem | Page No. |
| 01. | Traversing a two way linked list | 04-05 |
| 02. | Searching ITEM in a two way linked list | 06-07 |
| 03. | Deleting a node from a two way linked list | 08-09 |
| 04. | Inserting a node into a two way linked list | 10-12 |
| 05. | Adding an item into a stack (PUSH) | 13-14 |
| 06. | Deleting an item from a stack (POP) | 15-16 |
| 07. | <empty> |  |
| 08 | <empty> |  |
| 09 | <empty> |  |
| 10. | <empty> |  |

# Theory: A two way linked list is a linear collection of data elements,called **nodes**,where the linear order is given by means of pointers. Each node is divided into three parts :

# The first part contains the **information** of the elements.

# The second part called the **Forward field** or **next pointer field** that contains the address of the next node in the list.

# The third part called the **Backward field** or **previous pointer field** that contains the address of the previous node in the list.

# The operations normally performed on any linear structures are :

1. Traversal : Processing each elements in the list.
2. Search : Finding the location of an element.
3. Insertion : Adding a new element to the list.
4. Deletion : Removing an element from the list.

Etc.

A Stack is a linear structure in which items may be added or removed one by one only at the end. It means that the last item to be added to a stack is the first item to be removed. The main concept of stack is Last In - First Out.

# In a stack there is a pointer **TOP** that shows the **location** of **last data item.** There are two operations that are normally performed on any stack :

1. PUSH : Inserting an element into a stack.
2. POP : Deleting an element from a stack.

Problem No: 01

Problem Statement: Traversing a two way linked list.

Flow Chart :

START

PTR : START

PTR!=NULL

&& PTR > 0

yes no

PTR : FORW[PTR]

END

INFO[PTR]

Algorithm:

LIST is a two way linked list in memory. This algorithm traverses LIST, applying an operation PROCESS to each element of LIST. The variable PTR points to a node currently being processed.

1. Set PTR:= START

2. Repeat steps 3 and 4 while PTR≠NULL

3. Apply PROCESS to INFO[PTR]

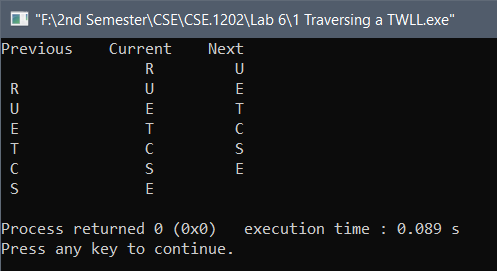
4. SET PTR := FORW[PTR]

[End of Repeat 2 loop]

5. Exit.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  char INFO[12]={'\0','\0','U','E','C','R','T','E','\0','\0','S','\0'};  int FORW[12]={12,9,8,'\0',11,3,5,7,'\0',2,4,10};  int BACK[12]={0,0,6,11,7,0,8,3,0,0,5,0};  int ptr,start=6,Back\_start=4,avail=1;    printf("Previous Current Next\n");  ptr=start;  while(ptr!='\0'){  printf(" %c\t\t%c\t %c\n",INFO[BACK[ptr-1]-1],INFO[ptr-1],INFO[FORW[ptr-1]-1]);  ptr=FORW[ptr-1];  }  return 0;  } |

Output:

Problem No: 02

Problem Statement: Searching ITEM in a two way linked list.

Flow Chart :

START

PTR : START

PTR!=NULL

&& PTR > 0

yes no

LOC : NULL

ITEM : INFO[PTR]

= =

LOC : PTR

!=

END

PTR : FOEW[PTR]

Algorithm: SEARCH (INFO, LINK, START, ITEM, LOC)

LIST is a two way linked list in memory. This algorithm finds the location LOC of the node where ITEM first appears in LIST, or sets LOC=NULL.

1. Set PTR:= START

2. Repeat steps 3 and 4 while PTR≠NULL

3. If ITEM = INFO[PTR] then:

Set LOC:=PTR and Exit.

Else:

SET PTR := FORW[PTR]

[End of If statement]

[End of Repeat 2 loop]

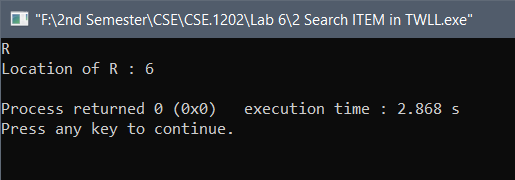
4. [Search is unsuccessful] Set LOC:=NULL

5. Exit.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  char INFO[12]={'\0','\0','U','E','C','R','T','E','\0','\0','S','\0'},ITEM;  int FORW[12]={12,9,8,'\0',11,3,5,7,'\0',2,4,10};  int BACK[12]={0,0,6,11,7,0,8,3,0,0,5,0};  int ptr,start=6,Back\_start=4,avail=1,LOC;    scanf("%c",&ITEM);  ptr=start;  while(INFO[ptr-1]!=ITEM&&ptr!='\0'){  ptr=FORW[ptr-1];  }  if(ptr=='\0')  printf("%c not found in INFO\n",ITEM);  else{  LOC=ptr;  printf("Location of %c : %d\n",ITEM,LOC);  }  return 0;  } |

Output:



Problem No: 03

Problem Statement: Deleting a node from a two way linked list.

Flow Chart :

START

LOC : START

yes no

FORW[BACK[LOC]] : FORW[LOC]

&&

BACK[FORW[LOC]]: BACK[LOC]

START : FORW[LOC]

&&

BACK[FORW[LOC]]: 0

FORW[LOC] : AVAIL

AVAIL: LOC

BACK[LOC] : 0

END

# **Algorithm: DEL (INFO, LINK, START, AVAIL, LOC, LOCP)**

This algorithm delete the node N with location LOC. When LOC is the first node, LOC = START.

1. IF LOC=START then:

Set START:= FORW[LOC] and

Set BACK[FORW[LOC]]**=**0 . [Delete First Node]

Else:

Set FORW[BACK[LOC]]=FORW[LOC] and

Set BACK[FORW[LOC]]=BACK[LOC] . [Delete N node]

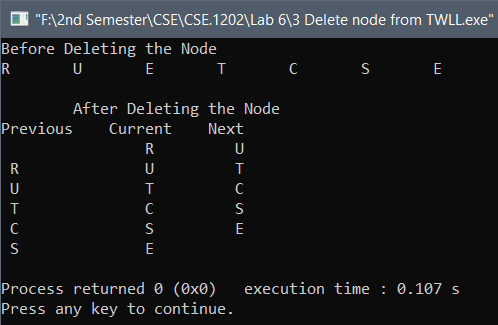
[End of IF structure]

2. FORW[LOC]=AVAIL and AVAIL=LOC and BACK[LOC-1]=0 .

**3.** Exit.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  char INFO[12]={'\0','\0','U','E','C','R','T','E','\0','\0','S','\0'},ITEM;  int FORW[12]={12,9,8,'\0',11,3,5,7,'\0',2,4,10};  int BACK[12]={0,0,6,11,7,0,8,3,0,0,5,0};  int ptr,start=6,Back\_start=4,avail=1,LOC=8;  printf("Before Deleting the Node\n");  ptr=start;  while(ptr!='\0'){  printf("%c\t",INFO[ptr-1]);  ptr=FORW[ptr-1]; }  if(LOC==start){  start=FORW[LOC-1];  BACK[FORW[LOC-1]-1]=0; }  else{  FORW[BACK[LOC-1]-1]=FORW[LOC-1];  BACK[FORW[LOC-1]-1]=BACK[LOC-1];  FORW[LOC-1]=avail;  avail=LOC;  BACK[LOC-1]=0; }  printf("\n\n\tAfter Deleting the Node\n");  printf("Previous Current Next\n");  ptr=start;  while(ptr!='\0'){  printf(" %c\t\t%c\t %c\n",INFO[BACK[ptr-1]-1],INFO[ptr-1],INFO[FORW[ptr-1]-1]);  ptr=FORW[ptr-1]; }  return 0;  } |

Output:

Problem No: 04

Problem Statement: Inserting a node into a two way linked list.

Flow Chart :

START

AVAIL : NULL

no yes

NEW : AVAIL

AVAIL : FORW [AVAIL]

OVERFLOW

INFO[NEW] : ITEM

LOC : NULL

= =

!=

START : NEW

:

FORW[NEW] : FORW[LOC]

&&

BACK[NEW]: LOC

FORW[START] : 0

BACK[START] : 0

BACK[FORW[LOC]]: NEW

&&

FORW[LOC] : NEW

END

# **Algorithm: INSTWL(INFO,FORW,BACK,START,AVAIL,LOC, ITEM)**

# 1. [OVERFLOW?] If AVAIL = NULL, then Write: OVERFLOW, and Exit

# 2. [Remove first node from AVAIL list]

# Set NEW := AVAIL, AVAIL:= FORW [AVAIL] Set INFO[NEW]:=ITEM.

# [Insert node into list]

If LOC=0 then

Set START=NEW, FORW[START]=0, BACK[START]=0.

Else

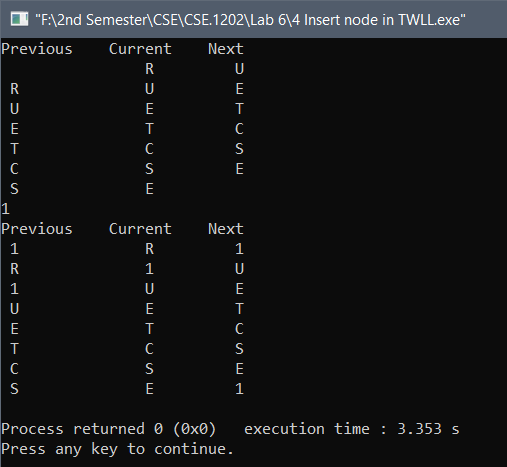
Set FORW[NEW] := FORW[LOC], BACK[NEW]:= LOC.

Set BACK[FORW[LOC]]:=NEW, FORW[LOC]:= NEW.

# 4. Exit.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  char INFO[12]={'\0','\0','U','E','C','R','T','E','\0','\0','S','\0'},ITEM;  int FORW[12]={12,9,8,'\0',11,3,5,7,'\0',2,4,10};  int BACK[12]={0,0,6,11,7,0,8,3,0,0,5,0};  int ptr,start=6,Back\_start=4,avail=1,LOC=8,New;  printf("Previous Current Next\n");  ptr=start;  while(ptr!='\0'){  printf(" %c\t\t%c\t %c\n",INFO[BACK[ptr-1]-1],INFO[ptr-1],INFO[FORW[ptr-1]-1]);  ptr=FORW[ptr-1]; }  if(avail=='\0')  printf("Overflow");  else{  New=avail;  avail=FORW[avail-1];  scanf("%c",&ITEM);  INFO[New-1]=ITEM;  if(LOC==0){ start=NEW; FORW[NEW]=0; BACK[NEW]=0; }  else{  FORW[New-1]=FORW[LOC-1];  BACK[New-1]=LOC;  BACK[FORW[LOC-1]-1]=New;  FORW[LOC-1]=New; }  printf("Previous Current Next\n");  ptr=start;  while(ptr!='\0'){  printf(" %c\t\t%c\t %c\n",INFO[BACK[ptr-1]-1],INFO[ptr-1],INFO[FORW[ptr-1]-1]);  ptr=FORW[ptr-1]; }  return 0;  } |

Output:

Problem No: 05

Problem Statement: Adding an item into a stack (PUSH).

Flow Chart :

START

TOP : MAXSTK

no yes

OVERFLOW

TPO : TOP+1

STACK[TPO] : ITEM

END

# **Algorithm:** **PUSH(STACK, TOP, MAXSTK, ITEM)**

# This procedure pushes an ITEM into a stack.

# 1. [Stack already filled]

# IF TOP = MAXSTK, then Write: OVERFLOW, and Return

# 2. Set TOP:= TOP+1.

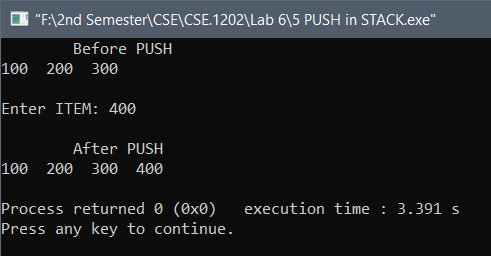
# 3. Set STACK[TOP]:= ITEM.

# 4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  int STACK[10]={100,200,300};  int top=3,max=10,ITEM,i;  printf("\tBefore PUSH\n");  for(i=0;i<top;i++)  printf("%d ",STACK[i]);  if(top==max)  printf("Overflow\n");  else{  printf("\n\nEnter ITEM: ");  scanf("%d",&ITEM);  top=top+1;  STACK[top-1]=ITEM;  }  printf("\n\tAfter PUSH\n");  for(i=0;i<top;i++)  printf("%d ",STACK[i]);  printf("\n");  return 0;  } |

Output:



Problem No: 06

Problem Statement: Deleting an item from a stack (POP).

Flow Chart :

START

TOP : 0

no yes

ITEM : STACK[TPO]

OVERFLOW

TPO : TOP-1

END

# **Algorithm:** **POP(STACK, TOP, ITEM)**

# This procedure deletes the top elements of STACK and assigns it to the variable ITEM.

# 1. [Stack already Empty]

# IF TOP = 0 then: Write: UNDERFLOW, and Return.

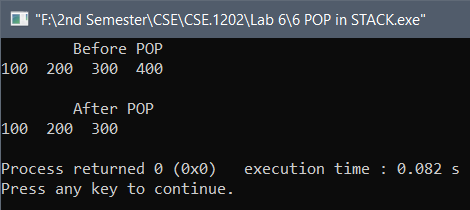
# 2. Set ITEM:= STACK[TOP]

# 3. Set TOP:= TOP-1

# 4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  int STACK[10]={100,200,300};  int top=3,max=10,ITEM,i;  printf("\tBefore POP\n");  for(i=0;i<top;i++)  printf("%d ",STACK[i]);  if(top==max)  printf("Overflow\n");  else{  top=top+1;  STACK[top-1]=ITEM;  }  printf("\n\tAfter POP\n");  for(i=0;i<top;i++)  printf("%d ",STACK[i]);  printf("\n");  return 0;  } |

Output:

# THE END #