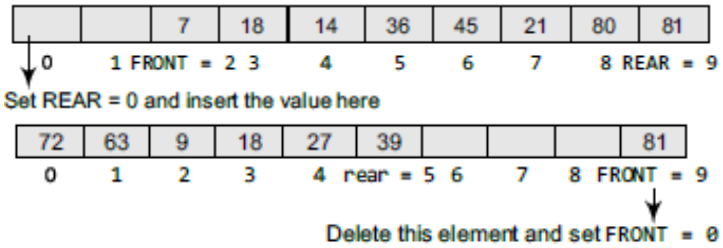
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2017-2018
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ
ΜΑΘΗΜΑ: ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΤΗΡΙΟ 6 – ΣΗΜΕΙΩΣΕΙΣ

Κυκλικές Ουρές

Μετά την εκτέλεση ενεργειών πάνω σε μια ουρά (υλοποίηση με πίνακα) προκύπτουν θέσεις στην αρχή του πίνακα οι οποίες μένουν κενές. Υπάρχουν δύο λύσεις για αυτές τις θέσεις: είτε μετακινούμε τα στοιχεία προς τα αριστερά (την αρχή του πίνακα) ή υιοθετούμε κυκλικές ουρές.



Μια κυκλική ουρά θα είναι γεμάτη μόνο όταν FRONT=0 & REAR=max-1 (max είναι το μέγεθος του πίνακα). Ακολουθούν οι αλγόριθμοι εισαγωγής και εξαγωγής στοιχείων.

```
Step 1: IF FRONT = 0 and Rear = MAX - 1
            Write "OVERFLOW"
            Goto step 4
        [End OF IF]
Step 2: IF FRONT = -1 and REAR = -1
            SET FRONT = REAR = 0
        ELSE IF REAR = MAX - 1 and FRONT != 0
            SET REAR = 0
        ELSE
            SET REAR = REAR + 1
        [END OF IF]
Step 3: SET QUEUE[REAR] = VAL
Step 4: EXIT
```

```
Step 1: IF FRONT = -1
            Write "UNDERFLOW"
            Goto Step 4
       [END of IF]
Step 2: SET VAL = QUEUE[FRONT]
Step 3: IF FRONT = REAR
            SET FRONT = REAR = -1
        ELSE
            IF FRONT = MAX -1
                  SET FRONT = 0
            ELSE
                  SET FRONT = FRONT + 1
            [END of IF]
       [END OF IF]
Step 4: EXIT
```

Code 1

(Υλοποιεί με τη χρήση ξεχωριστών συναρτήσεων όλες τις λειτουργίες σε μια κυκλική ουρά – υλοποίηση με πίνακα)

```
#include <stdio.h>
#include <conio.h>
#define MAX 10

int queue[MAX];
int front=–1, rear=–1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);

int main()
{
        int option, val;
        do
        {
        printf("\n ***** MAIN MENU *****");
        printf("\n 1. Insert an element");
        printf("\n 2. Delete an element");
        printf("\n 3. Peek");
        printf("\n 4. Display the queue");
        printf("\n 5. EXIT");
        printf("\n Enter your option : ");
        scanf("%d", &option);
        switch(option)
        {
                case 1:
                        insert();
                        break;
                case 2:
                        val = delete_element();
                        if(val!=–1)
                                printf("\n The number deleted is : %d", val);
                        break;
                case 3:
                        val = peek();
                        if(val!=–1)
                                printf("\n The first value in queue is : %d", val);
                        break;
                case 4:
                        display();
                        break;
```

```
                }
        }while(option!=5);
        getch();
        return 0;
}

void insert()
{
        int num;
        printf("\n Enter the number to be inserted in the queue : ");
        scanf("%d", &num);
        if(front==0 && rear==MAX–1)
                printf("\n OVERFLOW");
        else if(front==–1 && rear==–1)
        {
                front=rear=0;
                queue[rear]=num;
        }
        else if(rear==MAX–1 && front!=0)
        {
                rear=0;
                queue[rear]=num;
        }
        else
        {
                rear++;
                queue[rear]=num;
        }
}

int delete_element()
{
        int val;
        if(front==–1 && rear==–1)
        {
                printf("\n UNDERFLOW");
                return –1;
        }
        val = queue[front];
        if(front==rear)
                front=rear=–1;
        else
        {
                if(front==MAX–1)
```

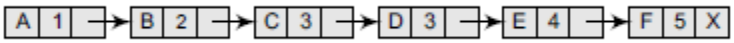
```
                front=0;
        else
                front++;
    }
    return val;
}

int peek()
{
    if(front==–1 && rear==–1)
    {
        printf("\n QUEUE IS EMPTY");
        return –1;
    }
    else
    {
        return queue[front];
    }
}

void display()
{
    int i;
    printf("\n");
    if (front ==–1 && rear= =–1)
        printf ("\n QUEUE IS EMPTY");
    else
    {
        if(front<rear)
        {
            for(i=front;i<=rear;i++)
                printf("\t %d", queue[i]);
        }
        else
        {
            for(i=front;i<MAX;i++)
                printf("\t %d", queue[i]);
            for(i=0;i<=rear;i++)
                printf("\t %d", queue[i]);
        }
    }
}
```

Ουρές με προτεραιότητα

Στις ουρές αυτές, το κάθε ένα στοιχείο λαμβάνει και μια προτεραιότητα που ορίζει τη σειρά επεξεργασίας του στοιχείου. Οι κανόνες που ισχύουν έχουν ως εξής:

- Τα στοιχεία με υψηλή προτεραιότητα επεξεργάζονται πριν από τα στοιχεία με χαμηλότερη προτεραιότητα.
- Δύο στοιχεία με ίδια προτεραιότητα επεξεργάζονται σε μια σειρά First-Come First-Served (FCFS) (ίδια με τη FIFO προσέγγιση).

Κατά την εξαγωγή, το στοιχείο με τη μεγαλύτερη προτεραιότητα βγαίνει πρώτο. Η υλοποίηση μιας ουράς με προτεραιότητα γίνεται είτε με μια ταξινομημένη λίστα ή με μια 'απλή' λίστα όπου οι εισαγωγές γίνονται στο τέλος της. Στην υλοποίηση με λίστα, ο κάθε κόμβος έχει τρία τμήματα: τα δεδομένα, την προτεραιότητα και το δείκτη προς τον επόμενο κόμβο.



Γενικά, μικρότερος αριθμός δείχνει μεγαλύτερη προτεραιότητα. Στην εισαγωγή, πρέπει να αναζητήσουμε το πρώτο κατά σειρά στοιχείο που έχει μικρότερη προτεραιότητα από το στοιχείο που πρόκειται να εισαχθεί. Ο νέος κόμβος εισάγεται πριν από τον κόμβο με τη μικρότερη προτεραιότητα. Αν υπάρχει στοιχείο με την ίδια προτεραιότητα, ο νέος κόμβος εισάγεται μετά από αυτόν. Η εξαγωγή είναι πιο εύκολη διαδικασία αφού το πρώτο στοιχείο της λίστας αφαιρείται από αυτή.

Code 2

(Υλοποιεί με τη χρήση ξεχωριστών συναρτήσεων όλες τις λειτουργίες σε μια ουρά με προτεραιότητα – υλοποίηση με λίστα)

```
#include <stdio.h>
#include <malloc.h>
#include <conio.h>

struct node
{
int data;
int priority;
struct node *next;
}

struct node *start=NULL;
struct node *insert(struct node *);
struct node *delete(struct node *);
void display(struct node *);

int main()
{
        int option;
        do
        {
                printf("\n *****MAIN MENU*****");
                printf("\n 1. INSERT");
```

```
                printf("\n 2. DELETE");
                printf("\n 3. DISPLAY");
                printf("\n 4. EXIT");
                printf("\n Enter your option : ");
                scanf( "%d", &option);
                switch(option)
                {
                        case 1:
                                start=insert(start);
                                break;
                        case 2:
                                start = delete(start);
                                break;
                        case 3:
                                display(start);
                                break;
                }
        }while(option!=4);
}

struct node *insert(struct node *start)
{
        int val, pri;
        struct node *ptr, *p;
        ptr = (struct node *)malloc(sizeof(struct node));
        printf("\n Enter the value and its priority : " );
        scanf( "%d %d", &val, &pri);
        ptr–>data = val;
        ptr–>priority = pri;
        if(start==NULL || pri < start–>priority )
        {
                ptr–>next = start;
                start = ptr;
        }
        else
        {
                p = start;
                while(p–>next != NULL && p–>next–>priority <= pri)
                        p = p–>next;
                ptr–>next = p–>next;
                p–>next = ptr;
        }
        return start;
}
```

```
struct node *delete(struct node *start)
{
	struct node *ptr;
	if(start == NULL)
	{
		printf("\n UNDERFLOW" );
		return;
	}
	else
	{
		ptr = start;
		printf("\n Deleted item is: %d", ptr–>data);
		start = start–>next;
		free(ptr);
	}
	return start;
}

void display(struct node *start)
{
	struct node *ptr;
	ptr = start;
	if(start == NULL)
		printf("\nQUEUE IS EMPTY" );
	else
	{
		printf("\n PRIORITY QUEUE IS : " );
		while(ptr != NULL)
		{
			printf( "\t%d[priority=%d]", ptr–>data, ptr–>priority );
			ptr=ptr–>next;
		}
	}
}
```

Στοίβες

Οι στοίβες αποτελούν σημαντικές δομές δεδομένων αφού κρατούν σε μια σειρά τα στοιχεία τα οποία αποθηκεύουν. Μια στοίβα είναι μια γραμμική δομή δεδομένων που υιοθετεί μια πολύ απλή αρχή: τα στοιχεία εισέρχονται ή εξέρχονται μόνο από ένα σημείο της στοίβας. Για αυτό το λόγο, μια στοίβα χαρακτηρίζεται από την αγγλική ορολογία Last In First Out – LIFO. Ένα χαρακτηριστικό παράδειγμα χρήσης μιας στοίβας είναι η κλήση συναρτήσεων σε ένα πρόγραμμα.

Ένας τρόπος αναπαράστασης μιας στοίβας είναι με τη βοήθεια πινάκων. Σχηματικά μια στοίβα έχει ως εξής:

A	AB	ABC	ABCD	ABCDE					
0	1	2	3	**TOP = 4**	5	6	7	8	9

Σε μια στοίβα μπορούμε να εκτελέσουμε δύο κύριες ενέργειες:

- Ώθηση (Push). Εισάγουμε ένα στοιχείο στη στοίβα.

```
Step 1: IF TOP = MAX-1
            PRINT "OVERFLOW"
            Goto Step 4
        [END OF IF]
Step 2: SET TOP = TOP + 1
Step 3: SET STACK[TOP] = VALUE
Step 4: END
```

- Απώθηση (Pop). Βγάζουμε ένα στοιχείο από τη στοίβα.

```
Step 1: IF TOP = NULL
            PRINT "UNDERFLOW"
            Goto Step 4
        [END OF IF]
Step 2: SET VAL = STACK[TOP]
Step 3: SET TOP = TOP - 1
Step 4: END
```

Σε κάθε περίπτωση, χρειαζόμαστε ένα δείκτη top ο οποίος μας δείχνει ποιο είναι το πάνω στοιχείο μιας στοίβας.

Code 3

(Υλοποιεί με τη χρήση ξεχωριστών συναρτήσεων όλες τις λειτουργίες σε μια στοίβα – υλοποίηση με πίνακα)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 3 // Altering this value changes size of stack created

int st[MAX], top=-1;
void push(int st[], int val);
int pop(int st[]);
int peek(int st[]);
void display(int st[]);

int main(int argc, char *argv[]) {
        int val, option;
        do
        {
                printf("\n *****MAIN MENU*****");
                printf("\n 1. PUSH");
```

```
		printf("\n 2. POP");
		printf("\n 3. PEEK");
		printf("\n 4. DISPLAY");
		printf("\n 5. EXIT");
		printf("\n Enter your option: ");
		scanf("%d", &option);
		switch(option)
		{
			case 1:
				printf("\n Enter the number to be pushed on stack: ");
				scanf("%d", &val);
				push(st, val);
				break;
			case 2:
				val = pop(st);
				if(val != -1)
					printf("\n The value deleted from stack is: %d", val);
				break;
			case 3:
				val = peek(st);
				if(val != -1)
					printf("\n The value stored at top of stack is: %d", val);
				break;
			case 4:
				display(st);
			break;
		}
	}while(option != 5);
	return 0;
}

void push(int st[], int val)
{
	if(top == MAX-1)
	{
		printf("\n STACK OVERFLOW");
	}
	else
	{
		top++;
		st[top] = val;
	}
}
```

```
int pop(int st[])
{
	int val;
	if(top == -1)
	{
		printf("\n STACK UNDERFLOW");
		return -1;
	}
	else
	{
		val = st[top];
		top--;
		return val;
	}
}

void display(int st[])
{
	int i;
	if(top == -1)
		printf("\n STACK IS EMPTY");
	else
	{
		for(i=top;i>=0;i--)
			printf("\n %d",st[i]);
		printf("\n"); // Added for formatting purposes
	}
}

int peek(int st[])
{
	if(top == -1)
	{
		printf("\n STACK IS EMPTY");
		return -1;
	}
	else
		return (st[top]);
}
```

Αντίστοιχα, μια στοίβα μπορεί να υλοποιηθεί με τη βοήθεια συνδεδεμένων λιστών. Το κάθε στοιχείο θα εισαχθεί στην αρχή της λίστας από την οποία θα εξάγονται μελλοντικά τα στοιχεία. Ακολουθούν, οι αλγόριθμοι για την εκτέλεση των λειτουργιών push και pop.

Λειτουργία push

```
Step 1: Allocate memory for the new
        node and name it as NEW_NODE
Step 2: SET NEW_NODE->DATA = VAL
Step 3: IF TOP = NULL
            SET NEW_NODE->NEXT = NULL
            SET TOP = NEW_NODE
        ELSE
            SET NEW_NODE->NEXT = TOP
            SET TOP = NEW_NODE
        [END OF IF]
Step 4: END
```

Λειτουργία pop

```
Step 1: IF TOP = NULL
            PRINT "UNDERFLOW"
            Goto Step 5
        [END OF IF]
Step 2: SET PTR = TOP
Step 3: SET TOP = TOP->NEXT
Step 4: FREE PTR
Step 5: END
```

Code 4

(Υλοποιεί με τη χρήση ξεχωριστών συναρτήσεων όλες τις λειτουργίες σε μια στοίβα – υλοποίηση με λίστα)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct stack
{
        int data;
        struct stack *next;
};

struct stack *top = NULL;
struct stack *push(struct stack *, int);
struct stack *display(struct stack *);
struct stack *pop(struct stack *);
int peek(struct stack *);

int main(int argc, char *argv[]) {
        int val, option;
        do
```

```
	{
		printf("\n *****MAIN MENU*****");
		printf("\n 1. PUSH");
		printf("\n 2. POP");
		printf("\n 3. PEEK");
		printf("\n 4. DISPLAY");
		printf("\n 5. EXIT");
		printf("\n Enter your option: ");
		scanf("%d", &option);
		switch(option)
		{
			case 1:
				printf("\n Enter the number to be pushed on stack: ");
				scanf("%d", &val);
				top = push(top, val);
				break;
			case 2:
				top = pop(top);
				break;
			case 3:
				val = peek(top);
				if (val != -1)
					printf("\n The value at the top of stack is: %d", val);
				else
					printf("\n STACK IS EMPTY");
				break;
			case 4:
				top = display(top);
				break;
		}
	}while(option != 5);
	return 0;
}

struct stack *push(struct stack *top, int val)
{
	struct stack *ptr;
	ptr = (struct stack*)malloc(sizeof(struct stack));
	ptr -> data = val;
	if(top == NULL)
	{
		ptr -> next = NULL;
		top = ptr;
	}
```

```
        else
        {
                ptr -> next = top;
                top = ptr;
        }
        return top;
}

struct stack *display(struct stack *top)
{
        struct stack *ptr;
        ptr = top;
        if(top == NULL)
                printf("\n STACK IS EMPTY");
        else
        {
                while(ptr != NULL)
                {
                        printf("\n %d", ptr -> data);
                        ptr = ptr -> next;
                }
        }
        return top;
}

struct stack *pop(struct stack *top)
{
        struct stack *ptr;
        ptr = top;
        if(top == NULL)
                printf("\n STACK UNDERFLOW");
        else
        {
                top = top -> next;
                printf("\n The value being deleted is: %d", ptr -> data);
                free(ptr);
        }
        return top;
}

int peek(struct stack *top)
{
        if(top==NULL)
                return -1;
```

```
        else
                return top ->data;
}
```

Πολλαπλές Στοίβες

Ως εφαρμογή της στοίβας, μπορούμε να έχουμε πολλαπλές στοίβες όταν τις υλοποιούμε με τη βοήθεια ενός πίνακα. Σχηματικά, οι δύο στοίβες θα έχουν ως εξής:

Προφανώς, ο πίνακας που υιοθετούμε θα πρέπει να έχει αρκετό μέγεθος ώστε να φιλοξενήσει τις στοίβες.

Code 5

(Υλοποιεί με τη χρήση ξεχωριστών συναρτήσεων όλες τις λειτουργίες σε πολλαπλές στοίβες – υλοποίηση με πίνακα)

```
#include <stdio.h>
#include <conio.h>
#define MAX 10

int stack[MAX],topA=–1,topB=MAX;

void pushA(int val)
{
        if(topA==topB–1)
                printf("\n OVERFLOW");
        else
        {
                topA+= 1;
                stack[topA] = val;
        }
}

int popA()
{
        int val;
        if(topA==–1)
        {
                printf("\n UNDERFLOW");
                val = –999;
        }
        else
        {
                val = stack[topA];
                topA—;
```

```
        }
        return val;
}

void display_stackA()
{
        int i;
        if(topA==–1)
                printf("\n Stack A is Empty");
        else
        {
                for(i=topA;i>=0;i—)
                        printf("\t %d",stack[i]);
        }
}

void pushB(int val)
{
        if(topB–1==topA)
                printf("\n OVERFLOW");
        else
        {
                topB –= 1;
                stack[topB] = val;
        }
}

int popB()
{
        int val;
        if(topB==MAX)
        {
                printf("\n UNDERFLOW");
                val = –999;
        }
        else
        {
                val = stack[topB];
                topB++;
        }
}

void display_stackB()
{
```

```
	int i;
	if(topB==MAX)
		printf("\n Stack B is Empty");
	else
	{
		for(i=topB;i<MAX;i++)
			printf("\t %d",stack[i]);
	}
}

void main()
{
	int option, val;
	do
	{
		printf("\n *****MENU*****");
		printf("\n 1. PUSH IN STACK A");
		printf("\n 2. PUSH IN STACK B");
		printf("\n 3. POP FROM STACK A");
		printf("\n 4. POP FROM STACK B");
		printf("\n 5. DISPLAY STACK A");
		printf("\n 6. DISPLAY STACK B");
		printf("\n 7. EXIT");
		printf("\n Enter your choice");
		scanf("%d",&option);
		switch(option)
		{
			case 1: printf("\n Enter the value to push on Stack A : ");
				scanf("%d",&val);
				pushA(val);
				break;
			case 2: printf("\n Enter the value to push on Stack B : ");
				scanf("%d",&val);
				pushB(val);
				break;
			case 3: val=popA();
				if(val!=–999)
					printf("\n The value popped from Stack A = %d",val);
				break;
			case 4: val=popB();
				if(val!=–999)
					printf("\n The value popped from Stack B = %d",val);
				break;
			case 5: printf("\n The contents of Stack A are : \n");
```

```
                                        display_stackA();
                                        break;
                        case 6: printf("\n The contents of Stack B are : \n");
                                        display_stackB();
                                        break;
                }
        }while(option!=7);
        getch();
}
```