"Angel Kanchev" University of Ruse

**Course work**

**on**

**Synthesis and Analysis of Algorithms**

Student: Marin Marian-Puiu

Group: Erasmus +

Date: 11.11.2019 Lecturer: Tsvetozar Georgiev

**Contents**

*1. Task* ................................................................................................................. 2

*2. Description of the solution* .............................................................................. 3

*3. Description of the used procedures and functions* .......................................... 4

*4. Screenshots from the program* ........................................................................ 6

*5. Source code* ..................................................................................................... 10

1. **Task**

Define a double linked list, each node of which has the following structure:

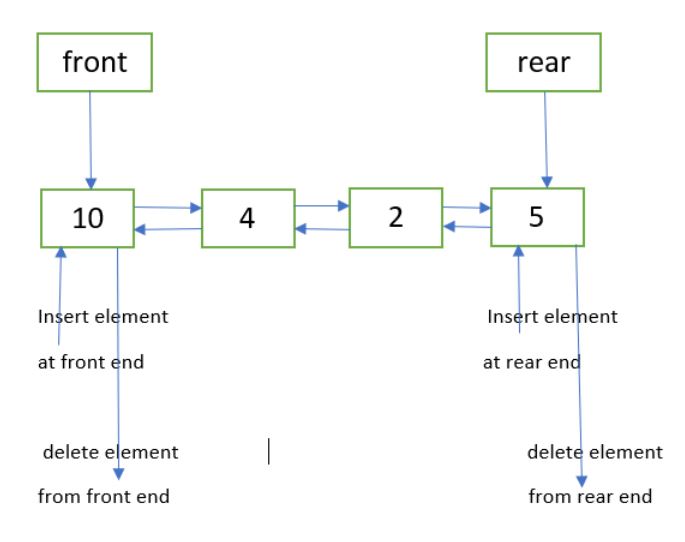
* Title of the book
* Author
* Language
* Publisher
* Year of publishing
* Number of pages
* Links ( to the next and to the previous node)

The program must perform the following operations:

* Select where to add node to the dequeue and add a node
* Select from where to remove a node from the dequeue and remove a node
* Display the number of nodes and the dequeue

***2. Description of the solution***

The program was created using **Visual Studio 2019** as an IDE. Each node has the following basic structure:



int Size = 0;

struct Nod {

string data;

string Titleofthebook;

string Author;

string Language;

string Typeofthebook;

int Year;

int Numberpages;

string Publisher;

Nod\* next;

Nod\* previous;

}\*inceput = NULL, \* sfarsit = NULL;

Every node is a structure that contains links to the previous and next elements of the list. By using the **inceput** and **sfarsit** variables we can easily add/remove new

nodes to/from the both sides of the list. The **size** variable keeps the

actual number of nodes, so it must be updated every time new node is added or

removed.

*I used comments in program in the program, to make it easier to understand!!*

***3. Description of the used procedures and***

***functions:***

**Function:** isEmpty()

**Description**: Function to check whether deque is empty or not.

**Parameters**: none.

**Called functions and procedures:** none.

**Function**: void insertInceput()

**Description**: Function to insert an element at the front end if the conditions are met.

**Parameters**: none.

**Called functions and procedures**: none.

**Function**: void insertSfarsit()

**Description**: Function to insert an element at the rear end if the conditions are met.

**Parameters**: none.

**Function**: deleteInceput()

**Description**: Function to delete an element at the front end if the conditions are met.

**Parameters**: none.

**Called functions and procedures**: none.

**Function**: deleteSfarsit()

**Description**: Function to delete an element at the rear end if the conditions are met.

**Parameters**: none.

**Called functions and procedures**: none.

**Function:** void Display()

**Parameters:** none

**Description:**

The Display() function simply goes through the whole dequeue, displaying each node’s information and the dequeue size.

**Function:** int main()

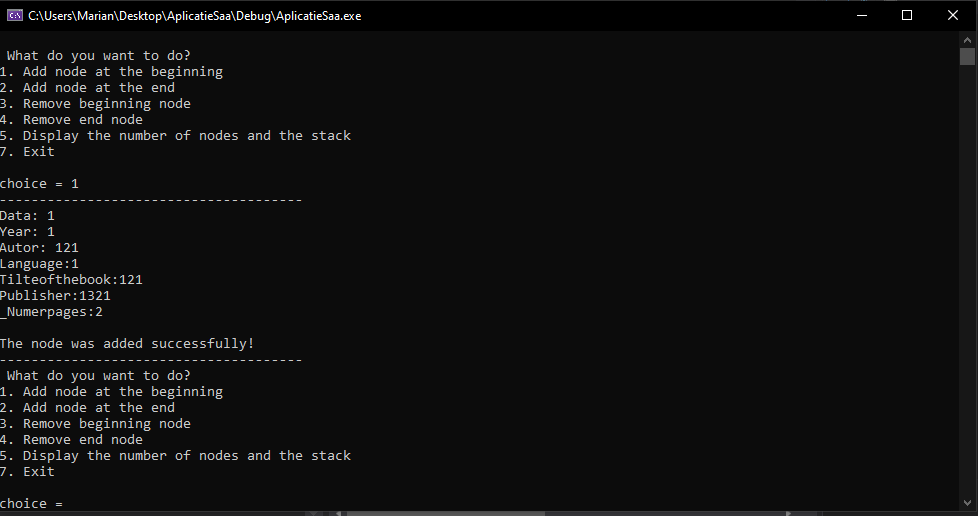
**Parameters:** none

**Description:**

I made a switch menu in the main() function that loops until the user selects the exit case.

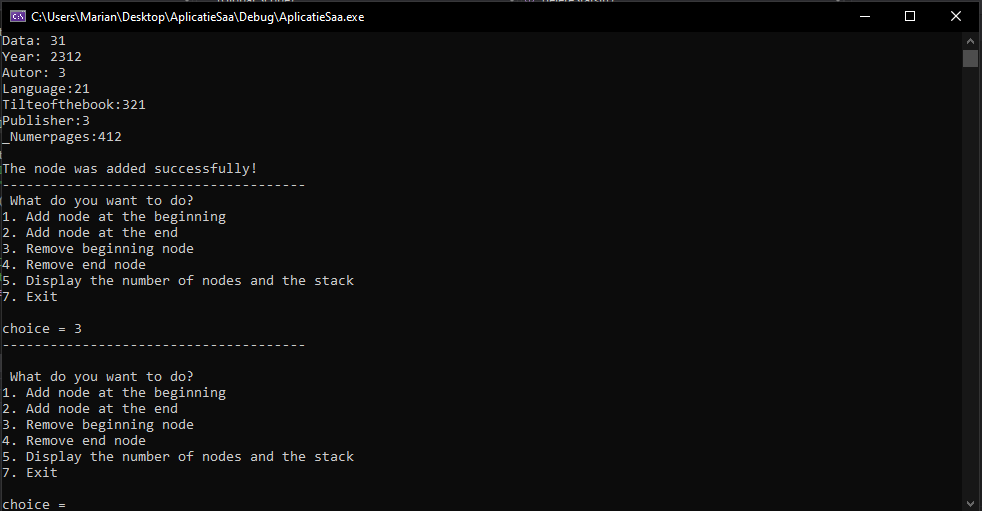
***4. Screenshots from the program***

**Add node in the front of the dequeue**





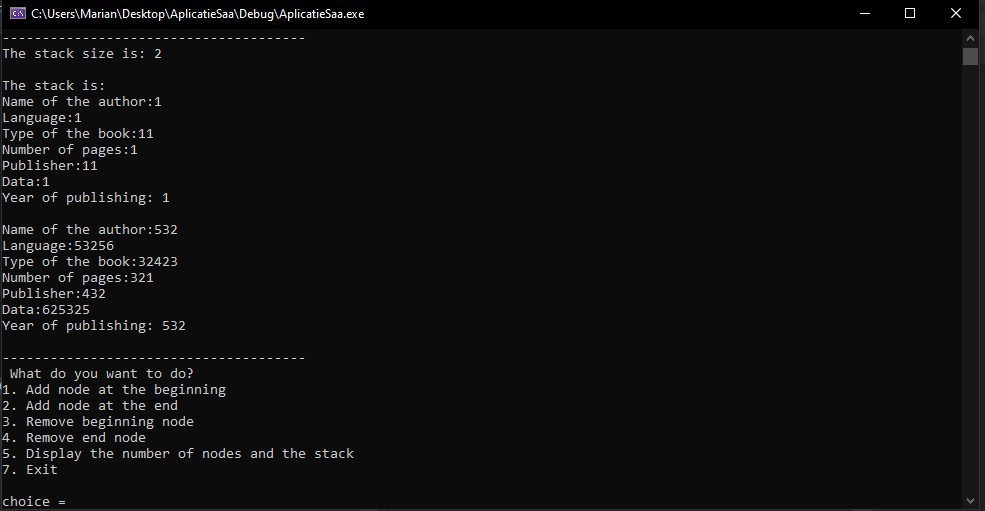
**Remove node at the front the dequeue**

****

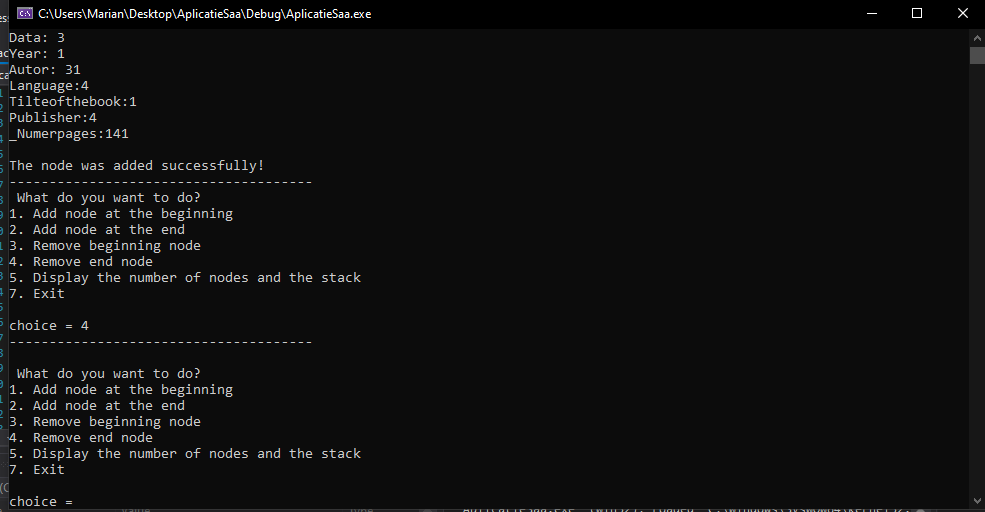
****

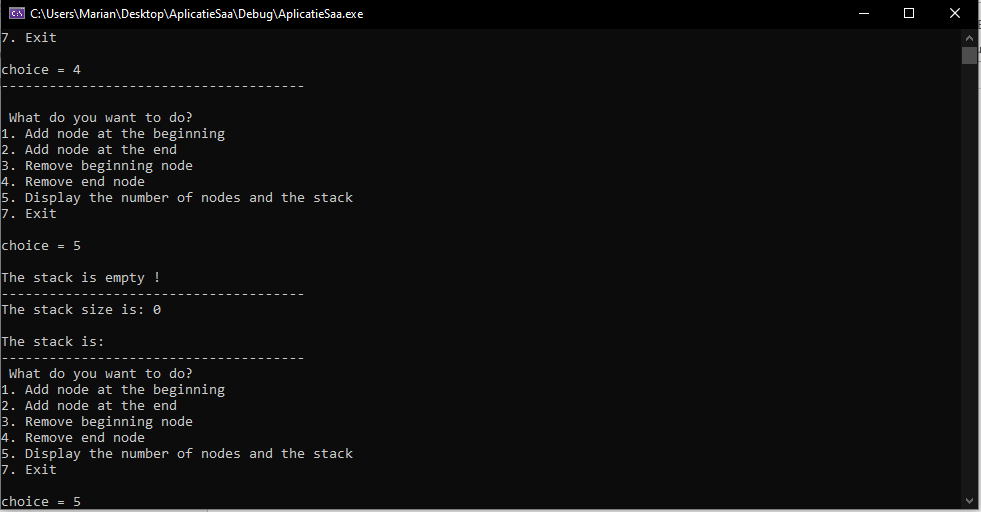
**Add node in the rear of the dequeue**

****

****

**Remove node at the rear from the dequeue**





**5. Source code**

// AplicatieSaa.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

// AplicatieSaa.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include <iostream>

using namespace std;

int Size = 0;

struct Nod {

string data;

string Titleofthebook;

string Author;

string Language;

string Typeofthebook;

int Year;

int Numberpages;

string Publisher;

Nod\* next;

Nod\* previous;

}\*inceput = NULL, \* sfarsit = NULL;

Nod\* newNode(string data, int \_Year, string \_Author, string \_Language, string \_Titleofthebook, string \_Publisher, int \_Numberpages) {

Nod\* nodNou = new Nod();

nodNou->data = data;

nodNou->Year = \_Year;

nodNou->Author = \_Author;

nodNou->Language = \_Language;

nodNou->Titleofthebook = \_Titleofthebook;

nodNou->Publisher = \_Publisher;

nodNou->Numberpages = \_Numberpages;

nodNou->previous = nodNou->next = NULL;

return nodNou;

}

bool isEmpty()

{

return (inceput == NULL);

}

int size()

{

return Size;

}

//Insert Front

void insertInceput()

{

cout << "--------------------------------------\n";

string data;

int \_Year;

string \_Author;

string \_Language;

string \_Titleofthebook;

string \_Publisher;

int \_Numberpages;

cout << "Data: ";

cin >> data;

cout << "Year: ";

cin >> \_Year;

cout << "Autor: ";

cin >> \_Author;

cout << "Language:";

cin >> \_Language;

cout << "Tilteofthebook:";

cin >> \_Titleofthebook;

cout << "Publisher:";

cin >> \_Publisher;

cout << "\_Numerpages:";

cin >> \_Numberpages;

Nod\* nodNou = newNode(data, \_Year, \_Author, \_Language, \_Titleofthebook, \_Publisher, \_Numberpages);

// If true then new element cannot be added

// and it is an 'Overflow' condition

if (nodNou == NULL)

cout << "Nu merge\n";

else

{

// If deque is empty

if (inceput == NULL)

sfarsit = inceput = nodNou;

// Inserts node at the front end

else

{

nodNou->next = inceput;

inceput->previous = nodNou;

inceput = nodNou;

}

// Increments count of elements by 1

Size++;

}

cout << "\nThe node was added successfully!\n";

cout << "--------------------------------------";

}

void insertSfarsit()

{

cout << "--------------------------------------\n";

// If true then new element cannot be added

// and it is an 'Overflow' condition

string data;

int \_Year;

string \_Author;

string \_Language;

string \_Titleofthebook;

string \_Publisher;

int \_Numberpages;

cout << "Data: ";

cin >> data;

cout << "Year: ";

cin >> \_Year;

cout << "Autor: ";

cin >> \_Author;

cout << "Language:";

cin >> \_Language;

cout << "Tilteofthebook:";

cin >> \_Titleofthebook;

cout << "Publisher:";

cin >> \_Publisher;

cout << "\_Numerpages:";

cin >> \_Numberpages;

Nod\* nouNod = newNode(data, \_Year, \_Author, \_Language, \_Titleofthebook, \_Publisher, \_Numberpages);

if (nouNod == NULL)

cout << "OverFlow\n";

else

{

// If deque is empty

if (inceput == NULL)

inceput = sfarsit = nouNod;

// Inserts node at the rear end

else

{

nouNod->previous = sfarsit;

sfarsit->next = nouNod;

sfarsit = nouNod;

}

Size++;

}

cout << "\nThe node was added successfully!\n";

cout << "--------------------------------------";

}

// Function to delete the element

// from the front end

void deleteInceput()

{

cout << "--------------------------------------\n";

// If deque is empty then

// 'Underflow' condition

if (isEmpty())

cout << "Nu se poate\n";

// Deletes the node from the front end and makes

// the adjustment in the links

else

{

Nod\* temp = inceput;

inceput = inceput->next;

// If only one element was present

if (inceput == NULL)

{

sfarsit = NULL;

Size--;

}

else {

inceput->previous = NULL;

free(temp);

// Decrements count of elements by 1

Size--;

cout << "\nThe node was removed successfully!\n";

cout << "--------------------------------------";

}

}

}

void Display()

{

//trebuie sa numar cate noduri sunt si sa afisez toata stiva

if (inceput == NULL)

{

cout << "\nThe stack is empty !\n";

}

else

cout << "--------------------------------------";

cout << "\nThe stack size is: " << Size << endl;

cout << "\nThe stack is: \n";

Nod\* aux = inceput;

while (inceput)

{

cout << "Name of the author:" << inceput->Author << '\n';

cout << "Language:" << inceput->Language << '\n';

cout << "Type of the book:" << inceput->Titleofthebook << '\n';

cout << "Number of pages:" << inceput->Numberpages << '\n';

cout << "Publisher:" << inceput->Publisher << '\n';

cout << "Data:" << inceput->data << '\n';

cout << "Year of publishing: " << inceput->Year << "\n\n";

inceput = inceput->next;

}

inceput = aux;

cout << "--------------------------------------";

}

void deleteSfarsit()

{

cout << "--------------------------------------\n";

// If deque is empty then

// 'Underflow' condition

if (isEmpty())

{

cout << "UnderFlow\n";

}

// Deletes the node from the rear end and makes

// the adjustment in the links

else

{

Nod\* temp = sfarsit;

sfarsit = sfarsit->previous;

// If only one element was present

if (sfarsit == NULL)

inceput = NULL;

else

{

sfarsit->next = NULL;

free(temp);

// Decrements count of elements by 1

}

Size--;

}

}

int main()

{

int choice;

while (1) {

cout << "\n What do you want to do?\n";

cout << "1. Add node at the beginning\n";

cout << "2. Add node at the end\n";

cout << "3. Remove beginning node\n";

cout << "4. Remove end node\n";

cout << "5. Display the number of nodes and the stack\n";

cout << "7. Exit\n";

cout << "\nchoice = "; cin >> choice;

switch (choice)

{

case 1: insertInceput();

break;

case 2: insertSfarsit();

break;

case 3: deleteInceput();

break;

case 4: deleteSfarsit();

break;

case 5: Display();

break;

case 7: return 0;

default: return 0;

}

}

}