ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

# ПОДАТОЧНИ СТРУКТУРИ И ПРОГРАМИРАЊЕ 2023/2024

## Аудиториски вежби 5

## Редови на чекање (во C++)

## ЗАДАЧИ:

1. На шалтерот на МВР се чека за вадење на лични карти, пасоши и возачки дозволи. Секој човек, за кој се знае името, презимето и информација на кои шалтери треба да чека, застанува во редицата за исправата која ја вади. Ако треба да вади повеќе исправи прво вади лична карта, па пасош, па возачка дозвола (откако ќе биде услужен во приоритетната редица, застанува во друга редица). Кога се отвора шалтерот, најпрво се опслужуваат луѓето кои вадат лични карти, па пасоши, па возачки дозволи. Да се напише програма која го симулира гореопишаното дејство.

```
#include <iostream>
#include <string>
#include <stdlib.h> //zaradi exit -1
using namespace std;
#define MAX 50
#define MAX2 1000

struct chovek
{
	string ime, prezime;
	bool lk, p, vd;
};
struct element
{
	chovek info;
};

struct Queue
{
	element array[MAX];
	int front, rear;
	void init();
	bool isEmpty();
	bool isFull();
	void enqueue(element e);
	element dequeue();
	element peek();
};

void Queue::init()
{
	front = 0;
	rear = -1;
}

bool Queue::isFull()
{
	return (rear == MAX - 1);
}

bool Queue::isEmpty()
{
	return (rear == -1);
}

void Queue::enqueue(element e)
{
	if (isFull())
	{
		cout << "Nema mesto" << endl;
		return;
	}
	array[++rear] = e;
}
```

```
element Queue::dequeue()
{
	if (isEmpty())
	{
		cout << "Redot e prazen" << endl;
		exit(-1);
	}
	element pom = array[front];
	for (int i = front; i < rear; i++)
		array[i] = array[i + 1];
	rear--;
	return pom;
}

element Queue::peek()
{
	if (isEmpty())
	{
		cout << "Redot e prazen" << endl;
		exit(-1);
	}
	return array[front];
}

void opsluzhuvanje(Queue &r1, Queue &r2, Queue &r3)
{
	element e;
	while (!r1.isEmpty())
	{
		e = r1.dequeue();
		e.info.lk = 0;
		cout << "Vo redot za lichni karti opsluzhen e " << e.info.ime << " " <<
e.info.prezime << endl;
		if (e.info.p == 1) //ako treba da cheka vo druga redica go smestuvame, po
prioritet na redici
			r2.enqueue(e);
		else if (e.info.vd == 1)
			r3.enqueue(e);
	}

	while (!r2.isEmpty())
	{
		e = r2.dequeue();
		e.info.p = 0;
		cout << "Vo redot za pasoshi opsluzhen e " << e.info.ime << " " <<
e.info.prezime << endl;
		if (e.info.vd == 1) //ako treba da cheka vo redica za vozachki dozvoli, go
smestuvame tamu
			r3.enqueue(e);
	}
	while (!r3.isEmpty())
	{
		e = r3.dequeue();
		e.info.vd = 0;
		cout << "Vo redot za vozachki dozvoli opsluzhen e " << e.info.ime << " " <<
e.info.prezime << endl;
	}
}

int main()
{
	Queue rlk, rp, rvd;
	rlk.init();
	rp.init();
	rvd.init();
```

```
	chovek chovek;
	char znak;
	for (int i = 0; ; i++)
	{
		cout << "Vnesete ime: ";
		cin >> chovek.ime;
		cout << "Vnesete prezime: ";
		cin >> chovek.prezime;
		cout << "Dali kje chekate vo redot za lichni karti? (1=da, 0=ne)-> ";
		cin >> chovek.lk;
		cout << "Dali kje chekate vo redot za pasoshi? (1=da, 0=ne)-> ";
		cin >> chovek.p;
		cout << "Dali kje chekate vo redot za vozachki dozvoli? (1=da, 0=ne)-> ";
		cin >> chovek.vd;
		element e;
		e.info = chovek;
		if (chovek.lk) rlk.enqueue(e);
		else
			if (chovek.p) rp.enqueue(e);
			else
				if (chovek.vd) rvd.enqueue(e);
		cout << "Ako nema povekje lugje pritisni ." << endl;
		cin >> znak;
		if (znak == '.') break;
		else continue;
	}
	opsluzhuvanje(rlk, rp, rvd);
}
```

2. Од тастатура се внесуваат N букви. Од внесените букви буквата ‘X’ се појавува точно два пати. Таа буква го означува почетокот и крајот на секвенцата од букви кои треба да се повторат уште еднаш. Ваша задача е на екран да ја прикажете конечната секвенца од сите внесени букви, така што делот кој треба да се повторува (од првото до второто појавување на буквата ‘X’) ќе го прикажете уште еднаш, но во обратен редослед од оној во кој се јавува во влезната секвенца.

**Забелешка**: дозволено е користење само на редови и магацини, не смее да се користат низи.

**Пример**: Влезна секвенца: A K M X V W Q X D P.
Излезна секвенца: A K M X V W Q X X Q W V X D P

```
#include <iostream>
#include <string>
#include <stdlib.h> //zaradi exit -1
using namespace std;
#define MAX 50

struct element
{
	char info;
};

struct Queue
{
	element array[MAX];
	int front, rear;
	void init();
	bool isEmpty();
	bool isFull();
	void enqueue(element e);
	element dequeue();
	element peek();
};

void Queue::init()
{
```

```
	front = 0;
	rear = -1;
}

bool Queue::isFull()
{
	return (rear == MAX - 1);
}

bool Queue::isEmpty()
{
	return (rear == -1);
}

void Queue::enqueue(element e)
{
	if (isFull())
	{
		cout << "Nema mesto!" << endl;
		return;
	}
	array[++rear] = e;
}

element Queue::dequeue()
{
	if (isEmpty())
	{
		cout << "Redot e prazen!" << endl;
		exit(-1);
	}
	element pom = array[front];
	for (int i = front; i < rear; i++)
		array[i] = array[i + 1];
	rear--;
	return pom;
}

element Queue::peek()
{
	if (isEmpty())
	{
		cout << "Redot e prazen!" << endl;
		exit(-1);
	}
	return array[front];
}

struct Stack
{
	element array[MAX];
	int top;
	void init();
	bool isEmpty();
	bool isFull();
	void push(element e);
	element pop();
	element peek();
};

void Stack::init()
{
	top = -1;
}
```

```
bool Stack::isEmpty()
{
	return(top == -1);
}

bool Stack::isFull()
{
	return (top == MAX - 1);
}


void Stack::push(element e)
{
	if (isFull())
	{
		cout << "Nema mesto!" << endl;
		return;
	}

	array[++top] = e;
}

element Stack::pop()
{
	if (isEmpty())
	{
		cout << "Magacinot e prazen!" << endl;
		exit(-1);
	}
	return array[top--];
}

element Stack::peek()
{
	if (isEmpty())
	{
		cout << "Magacinot e prazen!" << endl;
		exit(-1);
	}
	return array[top];
}

int main()
{
	Queue sekvenca;
	sekvenca.init();
	element e;
	char bukva;

	cout << "Vnesuvajte bukvi, za kraj vnesete ." << endl;

	while (true)
	{
		cin >> bukva;
		if (bukva == '.')
			break;
		e.info = bukva;
		sekvenca.enqueue(e);
	}

	Queue konechen; konechen.init();
	Stack prevrti; prevrti.init();

	while (!sekvenca.isEmpty())
	{
```

```
                element pom = sekvenca.dequeue();
                if (pom.info != 'X')
                        konechen.enqueue(pom);
                else
                {
                        konechen.enqueue(pom);
                        prevrti.push(pom);

                        do  /* izminuvame se dodeka ne ja najdeme vtorata X bukva */
                        {
                                pom = sekvenca.dequeue();
                                konechen.enqueue(pom);
                                prevrti.push(pom);

                        } while (pom.info != 'X');
                        /* posle toa vo redot gi vrakjame elementite od magacinot */
                        while (!prevrti.isEmpty())
                                konechen.enqueue(prevrti.pop());
                }
        }

        while (!konechen.isEmpty())
                cout << konechen.dequeue().info << ", ";
}
```

3. Да се напише функција која како аргументи добива два реда и треба да излез да врати нов ред кој ги комбинира влезните редови. Функцијата е наменета да ги распредели студентите кои истовремено полагаат втор колоквиум по ПСП и БиП во еден ред (влезните редови се наменети за секој од предметите ПСП и БиП). За секој студент се знае презиме, број на индекс (без година), испит кој го полага и број на освоени поени на прв колоквиум. Притоа во излезниот ред треба прво да ги постават студентите кои освоиле над 40 поени на првиот колоквиум по ПСП, па студентите кои освоиле над 50 поени на првиот колоквиум по БИП, по што треба да се постават останатите студенти по ПСП (кои имаат под 40 поени) и на крај останатите студенти по БиП (кои имаат под 50 поени). Доколку некој студент има под 30 поени да не се запишува во излезниот ред. Во функцијата може да се користат помошни редови, но користење на низи не е дозволено. По завршувањето на функцијата двата влезни реда треба да бидат празни. Дополнително во програмата да се дефинира структурата ред заедно со сите функции кои се потребни за функционирање на редот: за иницијализација, за проверка дали редот е полн и дали е празен, за внесување елементи, за изнесување на елементи, за гледање на елементот кој е напред во редот и за печатење на елементите без вадење на истите од редот.

**Пример:**

| **На влез:** | | **На излез:** |
|---|---|---|
| **ПСП ред:** | **БиП ред:** | **Мешан ред:** |
| **\|Naumoska 134 PSP 90\|** | **\|Koloski 12 BiP 60\|** | **\|Spaseski 236 BiP 40\|** |
| **\|Stamatovski 14 PSP 80\|** | **\|Spaseski 236 BiP 40\|** | **\|Bogoeska 146 PSP 39\|** |
| **\|Bogoeska 146 PSP 39\|** | **\|Skrceski 150 BiP 80\|** | **\|Koloski 12 BiP 60\|** |
| **\|Vanevski 76 PSP 20\|** | | **\|Skrceski 150 BiP 80\|** |
| **Во редот за ПСП прв внесен елемент е** | | **\|Naumoska 134 PSP 90\|** |
| **\|Vanevski 76 PSP 20\|** | | **\|Stamatovski 14 PSP 80\|** |

**Решение:**

```
#include <iostream>
#include <stdlib.h>

using namespace std;
```

```
#define MAX 10

struct element {
	string prezime;
	int brInd;
	string ispit;
	int brPoeni;
};


struct Queue {
	element array[MAX];
	int front, rear;
	void init();
	bool isEmpty();
	bool isFull();
	void enqueue(element e);
	element dequeue();
	element peek();
	void print();
};

void Queue::init() {
	front = 0;
	rear = -1;
}

bool Queue::isFull() {
	return (rear == MAX - 1);
}

bool Queue::isEmpty() {
	return (rear == -1);
}

void Queue::enqueue(element e) {
	if (isFull()) {
		cout << "Redot e poln." << endl;
		return;
	}
	array[++rear] = e;
}

element Queue::dequeue() {
	if (isEmpty()) {
		cout << "Redot e prazen." << endl;
		exit(-1);
	}

	element pom = array[front];
	for (int i = front; i < rear; i++) {
		array[i] = array[i + 1];
	}
	rear--;
	return pom;
}

element Queue::peek() {
	if (isEmpty()) {
		cout << "Redot e prazen." << endl;
		exit(-1);
	}

	return array[front];
}
```

```
void Queue::print(){
	for(int i = 0; i<=rear;i++){
		cout<<array[i].prezime<<" "<<array[i].brInd<<" "<<array[i].ispit<<"
"<<array[i].brPoeni<<endl;
	}

}

Queue preuredi(Queue &r1, Queue &r2) {
	Queue help1, help2;
	Queue izlez;
	help1.init();
	help2.init();
	izlez.init();
	element e1, e2;

	while (!r1.isEmpty()) {
		e1 = r1.dequeue();
		if(e1.brPoeni>40)
			izlez.enqueue(e1);
		else
			help1.enqueue(e1);
	}

	while (!r2.isEmpty()) {
			e2 = r2.dequeue();
			if(e2.brPoeni>50)
				izlez.enqueue(e2);
			else
				help2.enqueue(e2);
	}

	while (!help1.isEmpty()) {
			e1 = help1.dequeue();
			if(e1.brPoeni>=30)
				izlez.enqueue(e1);

	}

	while (!help2.isEmpty()) {
				e2 = help2.dequeue();
				if(e2.brPoeni>=30)
					izlez.enqueue(e2);
	}

	return izlez;
}

int main() {
	Queue r1, r2, r;

	r1.init();
	r2.init();

	r1.enqueue({"Vanevski", 76, "PSP", 20});
	r1.enqueue({"Bogoeska",  146, "PSP", 39});
	r1.enqueue({"Stamatovski", 14, "PSP", 80});
	r1.enqueue({"Naumoska", 134, "PSP", 90});

	r2.enqueue({"Skrceski", 150, "BiP", 80});
	r2.enqueue({"Spaseski", 236, "BiP", 40});
	r2.enqueue({"Koloski", 12, "BiP", 60});

	r = preuredi(r1, r2);
```

```
	r.print();

	return 0;
}
```

4. Да се напише програма во која ќе се работи со редови. Еден ред се состои од повеќе елементи. Во елементите основен податочен тип е цел број. Во првиот ред се чуваат разместени елементи. Во вториот ред се чуваат позициите на елементите во кои треба да биде првиот ред. Во излезниот ред, елементите од првиот ред треба да се наредат според редоследот даден во вториот ред. Во програмата е дозволена употреба на дополнителни редови. Да се дефинира структурата на еден ред со основни елементи во кој има цели броеви. Да се дефинираат и функциите потребни за правилна работа на редот- додавање вредност, повлекување вредност, читање на првата вредност во редот. Употребата на низи или магацини не е дозволена.

Влезни редови

**Прв ред**

1 2 3 4 5 6 <- позиции во редот

**5 8 3 1 4 9** <- вредности на елементите во редот

**Втор ред (новите позиции на првиот ред)**

**1 6 2 4 3 5**

**Излезен ред - после аплицирање на позициите од вториот ред**

**5 9 8 1 3 4**

**Решение:**

```
#include <iostream>
#include <string>
#include <stdlib.h> //zaradi exit -1
using namespace std;
#define MAX 50
struct element
{
	int info;
};
struct Queue
{
	element array[MAX];
	int front, rear;
	void init();
	bool isEmpty();
	bool isFull();
	void enqueue(element e);
	element dequeue();
	element peek();
};
void Queue::init()
{
	front = 0;
	rear = -1;
}
bool Queue::isFull()
```

```
{
	return (rear == MAX - 1);
}
bool Queue::isEmpty()
{
	return (rear == -1);
}
void Queue::enqueue(element e)
{
	if (isFull())
	{
		cout << "Nema mesto!" << endl;
		return;
	}
	array[++rear] = e;
}
element Queue::dequeue()
{
	if (isEmpty())
	{
		cout << "Redot e prazen!" << endl;
		exit(-1);
	}
	element pom = array[front];
	for (int i = front; i < rear; i++)
		array[i] = array[i + 1];
	rear--;
	return pom;
}
element Queue::peek()
{
	if (isEmpty())
	{
		cout << "Redot e prazen!" << endl;
		exit(-1);
	}
	return array[front];
}

element NajdiElementNaPozicija(Queue red, int pozicija) {
	//cout << red.peek().info;
	while (--pozicija) {
		red.dequeue();
	}
	return red.peek();
}

void proveri(Queue& red1, Queue& red2) {
	Queue nov_red;
	nov_red.init();
	while (!red2.isEmpty()) {
		nov_red.enqueue(NajdiElementNaPozicija(red1, red2.dequeue().info));
	}

	while (!nov_red.isEmpty()) {
		cout << nov_red.dequeue().info << " ";
	}
	cout << endl;
}

int main()
{
	Queue red1, red2;
	red1.init();
	red2.init();
```

```
	int a[6] = { 5,8,3,1,4,9 };
	for (int i = 0; i < 6; i++) {
		element e;
		e.info = a[i];
		red1.enqueue(e);
	}
	while (!red1.isEmpty()) {
		cout << red1.dequeue().info << " ";
	}
	for (int i = 0; i < 6; i++) {
		element e;
		e.info = a[i];
		red1.enqueue(e);
	}
	cout << endl;

	int b[6] = { 1,6,2,4,3,5 };
	for (int i = 0; i < 6; i++) {
		element e;
		e.info = b[i];
		red2.enqueue(e);
	}
	proveri(red1, red2);

}
```

5. Да се напише функција за одземање на два броја со помош на редови. Секој број е претставен преку редот. Секој елемент во редот е едноцифрена целобројна вредност, чија тежина во бројот е одредена според редоследот на елементот. Имено, првиот елемент има најмала тежина, последниот елемент најголема. Функцијата, која ги добива редовите од броевите како аргументи, треба да направи одземање на двата броја и резултатот да го смести во нов ред, кој ќе биде проследен во главната функција за печатење. Притоа, секогаш бројот запишан во ред 1 е поголем од бројот запишан во ред 2. Внимавајте на позајмувањето.

**Пример:**

Треба да се одземат броевите 1457 и 389.
R1: 7 <-> 5 <-> 4 <-> 1
R2: 9 <-> 8 <-> 3
Резултат: 8 <-> 6 <-> 0 <-> 1

**Решение:**

```
#include <iostream>
#include <stdlib.h>

using namespace std;

#define MAX 10

struct Queue {
    int array[MAX];
    int front, rear;
    void init();
    bool isEmpty();
    bool isFull();
    void enqueue(int e);
    int dequeue();
```

```
    int peek();
    int length();
};

int Queue::length(){
    return rear - front + 1;
}

void Queue::init() {
    front = 0;
    rear = -1;
}

bool Queue::isFull() {
    return (rear == MAX - 1);
}

bool Queue::isEmpty() {
    return (rear == -1);
}

void Queue::enqueue(int e) {
    if (isFull()) {
        cout << "Redot e poln." << endl;
        return;
    }
    array[++rear] = e;
}

int Queue::dequeue() {
    if (isEmpty()) {
        cout << "Redot e prazen." << endl;
        exit(-1);
    }

    int pom = array[front];
    for (int i = front; i < rear; i++) {
        array[i] = array[i + 1];
    }
    rear--;
    return pom;
}

int Queue::peek() {
    if (isEmpty()) {
        cout << "Redot e prazen." << endl;
        exit(-1);
    }

    return array[front];
}

void odzemi(Queue r1, Queue r2, Queue *r) {
```

```
    Queue help;
    help.init();
    int e1, e2, result = 0, borrow = 0;

    while (!r2.isEmpty()) {
        e1 = r1.dequeue();
        e2 = r2.dequeue();
        cout << e1 << " " << e2 << endl;
        if (e1 >= e2) {
            result = e1 - e2;
            cout << result << endl;
        } else {
            borrow = e1 + 10;
            result = borrow - e2;
            int size = r1.length();
            while (r1.peek() == 0) {
                help.enqueue(9);
                r1.dequeue();
                size--;
            }
            while (!help.isEmpty()) {
                r1.enqueue(help.dequeue());
            }
            r1.enqueue(r1.dequeue() - 1);
            size--;
            while(size){
                r1.enqueue(r1.dequeue());
                size--;
            }
        }

        r->enqueue(result);
    }

    while (!r1.isEmpty()) {
        r->enqueue(r1.dequeue());
    }
}

int main() {
    Queue r1, r2, r;
    r.init();
    r1.init();
    r2.init();

    r1.enqueue(7);
    r1.enqueue(5);
    r1.enqueue(4);
    r1.enqueue(1);

    r2.enqueue(9);
    r2.enqueue(8);
    r2.enqueue(3);
```

```
    odzemi(r1, r2, &r);
    cout << "Razlikata e: " << endl;
    while (!r.isEmpty()) {
        cout << r.dequeue();
    }

    return 0;
}
```

6. Да се напише функција која како аргумент прима ред. За секој од елементите на овој ред функцијата ќе го одреди првиот следен елемент во редот кој по вредност е поголем од разгледуваниот елемент и таа вредност ќе ја додаде во нов ред. Добиените вредности за секој од елементите во редот се запишуваат во втор ред кој функцијата треба да го врати во главната програма за печатење. Притоа, доколку не постои вредност поголема од разгледуваниот елемент или пак за последниот елемент во редот (кој веќе нема следбеници) се внесува вредност -1.
Напишете ја структурата за ред заедно со функции за креирање на ред, додавање и вадење на елемент од ред и печатење на ред.

**Пример:**
R1: 5 <-> 4 <-> 6 <-> 2 <-> 3 <-> 10 <-> 8 <-> 9
Резултат: 6 <-> 6 <-> 10 <-> 3 <-> 10 <-> -1 <-> 9 <-> -1

**Решение:**

```
#include <iostream>
#include <stdlib.h>

using namespace std;

#define MAX 10

struct Queue {
	int array[MAX];
	int front, rear;
	void init();
	bool isEmpty();
	bool isFull();
	void enqueue(int e);
	int dequeue();
	int peek();
	int print();
};

void Queue::init() {
	front = 0;
	rear = -1;
}

bool Queue::isFull() {
	return (rear == MAX - 1);
}

bool Queue::isEmpty() {
	return (rear == -1);
}

void Queue::enqueue(int e) {
	if (isFull()) {
```

```
		cout << "Redot e poln." << endl;
		return;
	}
	array[++rear] = e;
}

int Queue::dequeue() {
	if (isEmpty()) {
		cout << "Redot e prazen." << endl;
		exit(-1);
	}

	int pom = array[front];
	for (int i = front; i < rear; i++) {
		array[i] = array[i + 1];
	}
	rear--;
	return pom;
}

int Queue::peek() {
	if (isEmpty()) {
		cout << "Redot e prazen." << endl;
		exit(-1);
	}

	return array[front];
}

void sledenpogolem(Queue r1, Queue *r2) {
	Queue help;
	help.init();
	int e1, change;

	while (!r1.isEmpty()) {
		e1 = r1.dequeue();
		cout << e1 << endl;
		change = 0;
		while(!r1.isEmpty() && !change) {
			if (e1 < r1.peek()) {
				r2->enqueue(r1.peek());
				change = 1;
				break;
			}
			help.enqueue(r1.dequeue());
		}
		if(!change) {
			r2->enqueue(-1);
		}
		while(!r1.isEmpty())
			help.enqueue(r1.dequeue());
		while(!help.isEmpty()) {
			r1.enqueue(help.dequeue());
		}
	}
}

int main() {
	Queue r1, r2;
	r1.init();
	r2.init();

	r1.enqueue(5);
	r1.enqueue(4);
	r1.enqueue(6);
```

```
	r1.enqueue(2);
	r1.enqueue(3);
	r1.enqueue(10);
	r1.enqueue(8);
	r1.enqueue(9);

	sledenpogolem(r1, &r2);

	while (!r2.isEmpty()) {
		cout << r2.dequeue() << "\t";
	}

	return 0;
}
```