Министерство образования Республики Беларусь

г. Минск

Государственное учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

**Лабораторная работа №6**

**“ ОЧЕРЕДЬ ”**

**Учебная группа 230501**

Выполнил: Кочеров Роман Сергеевич

Проверил: Селезнев Александр Игоревич

2023 год

**Вариант 4**

**Задание 1.** Создать очередь для целых (положительных и отрицательных) чисел. Максимальный размер очереди вводится с экрана. Создать функции для ввода, вывода и определения размера очереди. Ввести 6 элементов. Вывести элементы очереди до первого отрицательного (включительно). Вывести размер оставшейся очереди.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct FIFO

{

int data;

struct FIFO\* next;

struct FIFO\* prev;

};

struct FIFO\* create(struct FIFO\*, int);

void print(struct FIFO\*);

struct FIFO\* pop(struct FIFO\* head);

int vvod();

int main()

{

int i, x;

struct FIFO\* head;

head = NULL;

struct FIFO\* tail;

tail = NULL;

printf("Enter 6 elements:\n");

for (i = 0; i < 6; i++)

{

x = vvod();

tail = create(tail, x);

if (head == NULL)

head = tail;

}

printf("\n");

print(head);

free(head);

}

struct FIFO\* create(struct FIFO\* tail, int x) {

struct FIFO\* n;

n = (struct FIFO\*)malloc(sizeof(struct FIFO));

n->next = NULL;

n->prev = NULL;

n->data = x;

if (tail == NULL)

{

tail = n;

}

else

{

n->prev = tail;

tail->next = n;

tail = n;

}

return tail;

}

int vvod()

{

int var;

while ((scanf\_s("%d", &var) == 0) || getchar() != '\n')

{

printf("wrong input! try again\n");

rewind(stdin);

}

return var;

}

void print(struct FIFO\* head)

{

int flag = 0, ch = 0;

while ((head != NULL) && (flag == 0))

{

printf("%d ", head->data);

if (head->data < 0)

{

flag = 1;

}

ch++;

head = head->next;

}

printf("\nnumber of element's, that left in the queue: %d", 6 - ch);

printf("\n");

}

struct FIFO\* pop(struct FIFO\* head)

{

if (head != NULL)

{

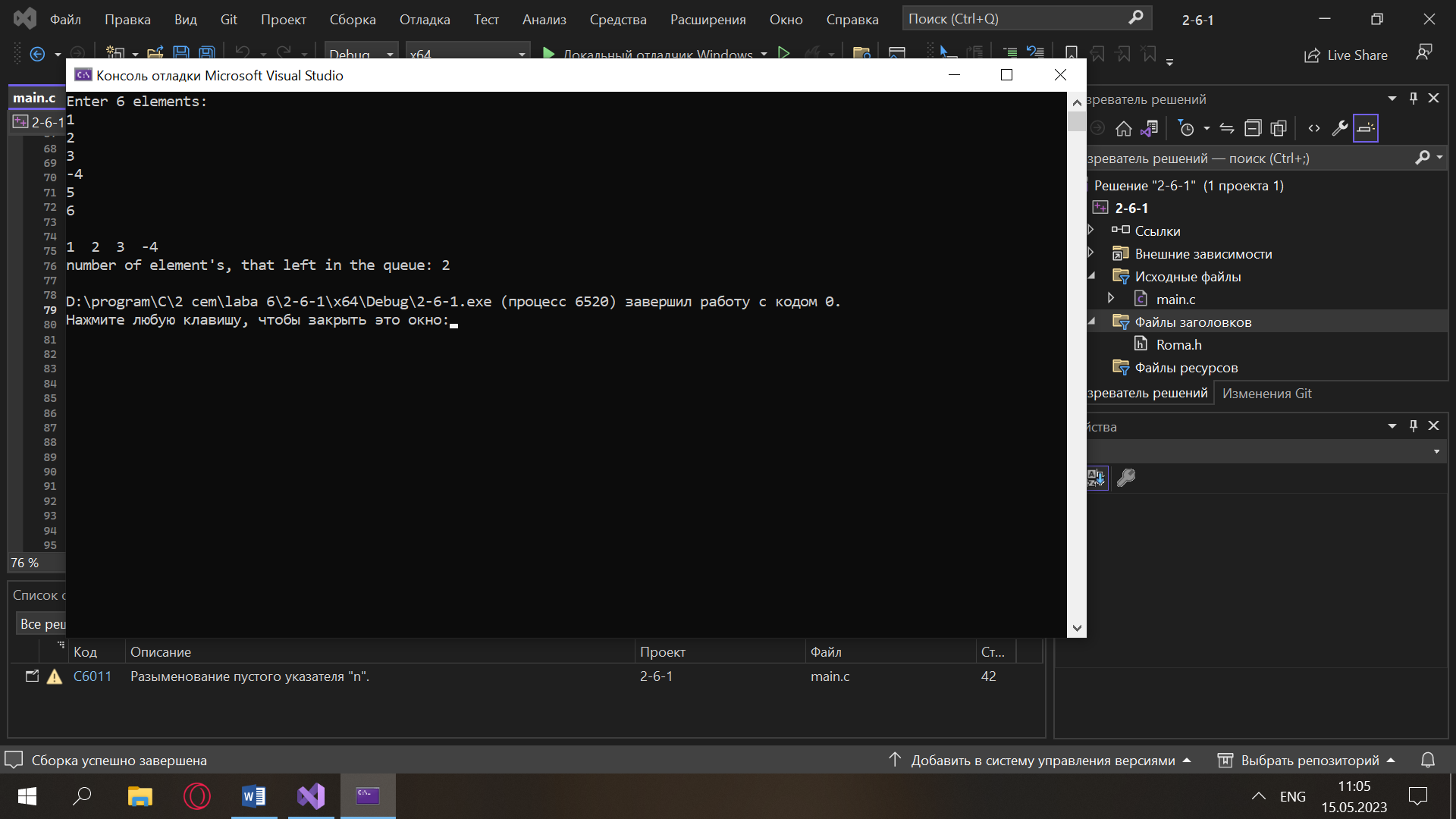
head = head->next;

return head;

}

else return NULL;

}



**Задание 2.** Создать дек для целых (положительных и отрицательных) чисел. Максимальный размер дека вводится с экрана. Создать функции для ввода, вывода и определения размера дека. Ввести 3 элемента справа и 3 слева. Вывести элементы дека справа до первого отрицательного (включительно). Вывести размер оставшегося дека.

#include <stdio.h>

#include <stdlib.h>

// определяем структуру для элемента очереди

struct QueueNode

{

int value;

struct QueueNode\* prev;

struct QueueNode\* next;

};

// определяем структуру для двунаправленной очереди

struct Deque

{

struct QueueNode\* head; // указатель на голову очереди

struct QueueNode\* tail; // указатель на хвост очереди

};

void enqueue\_front(struct Deque\* deque, int value);

void enqueue\_back(struct Deque\* deque, int value);

void dequeue\_front(struct Deque\* deque);

void dequeue\_back(struct Deque\* deque);

void print\_queue(struct Deque\* deque);

void romka(struct Deque\* deque);

int vvod();

int main() {

struct Deque deque;

deque.head = NULL;

deque.tail = NULL;

int buf;

for (int i = 0; i < 3; i++)

{

printf("Enter elemens on right: \n");

buf = vvod();

enqueue\_back(&deque, buf);

}

for (int i = 0; i < 3; i++)

{

printf("Enter elemens on left: \n");

buf = vvod();

enqueue\_front(&deque, buf);

}

romka(&deque);

return 0;

}

void enqueue\_front(struct Deque\* deque, int value)

{

struct QueueNode\* node = (struct QueueNode\*)malloc(sizeof(struct QueueNode));

node->value = value;

if (deque->head == NULL)

{

deque->head = node;

deque->tail = node;

node->prev = NULL;

node->next = NULL;

}

else

{

node->next = deque->head;

node->prev = NULL;

deque->head->prev = node;

deque->head = node;

}

}

// функция для добавления элемента в конец очереди

void enqueue\_back(struct Deque\* deque, int value) {

struct QueueNode\* node = (struct QueueNode\*)malloc(sizeof(struct QueueNode));

node->value = value;

if (deque->tail == NULL) {

deque->head = node;

deque->tail = node;

node->prev = NULL;

node->next = NULL;

}

else {

node->prev = deque->tail;

node->next = NULL;

deque->tail->next = node;

deque->tail = node;

}

}

// функция для удаления элемента с начала очереди

void dequeue\_front(struct Deque\* deque) {

if (deque->head == deque->tail) {

free(deque->head);

deque->head = NULL;

deque->tail = NULL;

}

else {

struct QueueNode\* temp = deque->head;

deque->head = deque->head->next;

deque->head->prev = NULL;

free(temp);

}

}

// функция для удаления элемента с конца очереди

void dequeue\_back(struct Deque\* deque) {

if (deque->head == deque->tail) {

free(deque->tail);

deque->head = NULL;

deque->tail = NULL;

}

else {

struct QueueNode\* temp = deque->tail;

deque->tail = deque->tail->prev;

deque->tail->next = NULL;

free(temp);

}

}

// функция для вывода на экран всех элементов очереди

void print\_queue(struct Deque\* deque) {

struct QueueNode\* current = deque->head;

while (current != NULL)

{

printf("%d ", current->value);

current = current->next;

}

printf("\n");

}

void romka(struct Deque\* deque) {

struct QueueNode\* current = deque->tail;

int block = 0, ch = 6;

while (current != NULL && block == 0)

{

if (current->value < 0)

{

block = 1;

}

ch--;

printf("%d ", current->value);

current = current->prev;

}

printf("\nnumber of elements left in deck: %d\n", ch);

}

int vvod()

{

int var;

while ((scanf\_s("%d", &var) == 0) || getchar() != '\n')

{

printf("wrong input! try again\n");

rewind(stdin);

}

return var;

}

