МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №3

**«Стэк и очередь»**

Выполнил:

Студент 1 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверила:

Войцехович О. Ю.

Брест 2022

**Цель работы:** Изучение принципов организации и работы с абстрактной структурой данных СТЭК списка и с абстрактной структурой данных ОЧЕРЕДЬ.

**Порядок выполнения работы**

*Задание 1*

Разработать консольное приложение, которое с помощью абстрактной структуры данных СТЭК проверяет соответствие открывающих и закрывающих HTML-тэгов во фрагменте HTML кода, введённого с клавиатуры.

**Реализация**

#include <iostream>

#include "stack.h"

using namespace std;

bool checkHTML();

int main()

{

string input = "y";

while (input == "y" || input == "") {

if (checkHTML()) {

cout << "HTML tag pairs are valid.\n";

}

else {

cout << "HTML tag pairs are invalid!\n";

}

cout << "Continue? (y/n)\n";

getline(cin, input); getline(cin, input);

}

return 0;

}

bool checkHTML() {

struct node\* tags = NULL; //stack

bool result = true;

cout << "Enter HTML code (to exit input mode, type 'e!'):\n";

char tmp;

cin >> noskipws >> tmp;

bool isTag = false;

bool isClosingTag = false;

bool isLastChrE = false;

bool exitCondition = false;

string tag;

while (!exitCondition) {

//request for exit

if (isLastChrE) {

if (tmp == '!') {

exitCondition = true;

continue;

}

isLastChrE = false;

}

if (tmp == 'e') {

isLastChrE = true;

}

//closing type of tag

if (isTag) {

if (tmp == '/') {

isClosingTag = true;

}

//end creating tag

else {

if (!isalpha(tmp) && !isdigit(tmp)) {

if (!isClosingTag) {

//push into stack if opening tag

push(&tags, tag);

}

else {

if (!isStackEmpty(tags) && top(tags) == tag) {

//pop from stack if pair matches

pop(&tags);

}

else {

result = false;

}

}

isTag = false;

isClosingTag = false;

}

//reading tag

else {

tag.push\_back(tmp);

}

}

}

//begin creating tag

if (tmp == '<') {

isTag = true;

tag = "";

}

cin >> noskipws >> tmp;

}

if (result && isStackEmpty(tags)) {

return true;

}

return false;

}

Файл **stack.h**

#include <string.h>

using namespace std;

struct node {

char value[255];

struct node\* next;

};

void push(struct node\*\* node, string value);

void pop(struct node\*\* node);

string top(struct node\* node);

bool isStackEmpty(struct node\* node);

void push(struct node\*\* node, string value) {

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

strcpy(tmp->value, value.c\_str());

tmp->next = (\*node);

(\*node) = tmp;

}

void pop(struct node\*\* node) {

if (isStackEmpty(\*node)) return;

(\*node) = (\*node)->next;

}

string top(struct node\* node) {

if (isStackEmpty(node)) return "";

return string(node->value);

}

bool isStackEmpty(struct node\* node) {;

if (node) return false;

return true;

}

*Задание 2.1*

Даны две непустые очереди, которые содержат одинаковое количество элементов. Объединить очереди в одну, в которой элементы исходных очередей чередуются.

**Реализация**

#include <iostream>

#include "queue.h"

using namespace std;

int main()

{

//We have 2 queues with elements (we shall define elements as int)

struct node\* queue1 = nullptr;

struct node\* queue2 = nullptr;

struct node\* queueOut = nullptr;

for (int i=0; i<20; i++) {

if (i<10) {

//0 1 2 3...

push(&queue1, i);

}

else {

//10 11 12 13...

push(&queue2, i);

}

}

for (int i=0; i<20; i++) {

if (!(i%2)) {

push(&queueOut, front(queue1));

pop(&queue1);

}

else {

push(&queueOut, front(queue2));

pop(&queue2);

}

}

for (int i=0; i<20; i++) {

cout << front(queueOut) << " ";

pop(&queueOut);

}

cout << endl;

return 0;

}

*Задание 2.2*

Даны две непустые очереди. Элементы каждой из очередей упорядочены по возрастанию. Объединить очереди в одну с сохранением упорядоченности элементов.

**Реализация**

#include <iostream>

#include "queue.h"

using namespace std;

int main()

{

//We have 2 queues with elements (we shall define elements as int)

struct node\* queue1 = nullptr;

struct node\* queue2 = nullptr;

struct node\* queueOut = nullptr;

for (int i=0; i<20; i++) {

if (!(i%3)) {

//0 3 6 9 12 15 18

push(&queue1, i);

}

else {

//1 2 4 5 7 8 10 11 13 14 16 17 19

push(&queue2, i);

}

}

int minimum;

while (!empty(queue1) || !empty(queue2)) {

if (front(queue1) < front(queue2)) {

minimum = front(queue1);

pop(&queue1);

}

else {

minimum = front(queue2);

pop(&queue2);

}

push(&queueOut, minimum);

if (empty(queue1)) {

while (!empty(queue2)) {

push(&queueOut, front(queue2));

pop(&queue2);

}

}

if (empty(queue2)) {

while (!empty(queue1)) {

push(&queueOut, front(queue1));

pop(&queue1);

}

}

}

//0 1 2 3 4 5...

for (int i=0; i<20; i++) {

cout << front(queueOut) << " ";

pop(&queueOut);

}

cout << endl;

return 0;

}

Файл **queue.h**

#include <string.h>

#include <iostream>

using namespace std;

struct node {

int value;

struct node\* next;

};

void push(struct node\*\* node, int value);

void pop(struct node\*\* node);

int front(struct node\* node);

bool empty(struct node\* node);

void push(struct node\*\* head, int value) {

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

tmp->value = value;

tmp->next = nullptr;

struct node\* node = \*head;

if (node) {

while (node->next) {

node = node->next;

}

node->next = tmp;

}

else {

(\*head) = tmp;

}

}

void pop(struct node\*\* node) {

\*node = (\*node)->next;

}

int front(struct node\* node) {

if (node) return node->value;

return NULL;

}

bool empty(struct node\* node) {

if (node) return false;

return true;

}

**Вывод:** Изучены и использованы принципы организации абстрактных структур данных СТЭК и ОЧЕРЕДЬ для реализаций задач, предполагающих их использование.