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>

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() {

stack<string> tags;

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

tags.push(tag);

}

else {

if (!tags.empty() && tags.top() == tag) {

//pop from stack if pair matches

tags.pop();

}

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 && tags.empty()) {

return true;

}

return false;

}

*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>

using namespace std;

int main()

{

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

queue<int> queue1;

queue<int> queue2;

queue<int> queueOut;

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

if (i<10) {

//0 1 2 3...

queue1.push(i);

}

else {

//10 11 12 13...

queue2.push(i);

}

}

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

if (!(i%2)) {

queueOut.push(queue1.front());

queue1.pop();

}

else {

queueOut.push(queue2.front());

queue2.pop();

}

}

//0 10 1 11 2 12...

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

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

queueOut.pop();

}

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>

using namespace std;

int main()

{

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

queue<int> queue1;

queue<int> queue2;

queue<int> queueOut;

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

if (!(i%3)) {

//0 3 6 9 12 15 18

queue1.push(i);

}

else {

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

queue2.push(i);

}

}

int minimum;

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

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

minimum = queue1.front();

queue1.pop();

}

else {

minimum = queue2.front();

queue2.pop();

}

queueOut.push(minimum);

if (queue1.empty()) {

while (!queue2.empty()) {

queueOut.push(queue2.front());

queue2.pop();

}

}

if (queue2.empty()) {

while (!queue1.empty()) {

queueOut.push(queue1.front());

queue1.pop();

}

}

}

//0 1 2 3 4 5...

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

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

queueOut.pop();

}

cout << endl;

return 0;

}

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