МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ КОЗЬМЫ МИНИНА»

**Факультет естественных математических и компьютерных наук**

**03.03.09 Прикладная информатика**

**Кафедра прикладной информатики и информационных технологий в образовании**

**Специальность Прикладная информатика в менеджменте**

**Лабораторная №1**

**«Динамические структуры данных»**

Работу выполнил:

Студент 2 курса д/о группы ПИМ-18

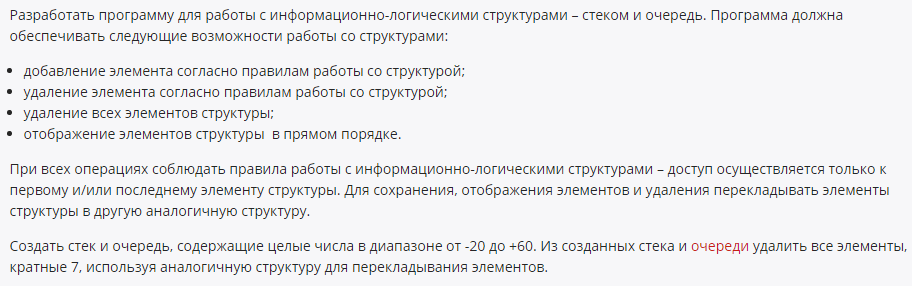
Власов Владимир

Работу проверил:

Старший преподаватель Круподерова К.Р.

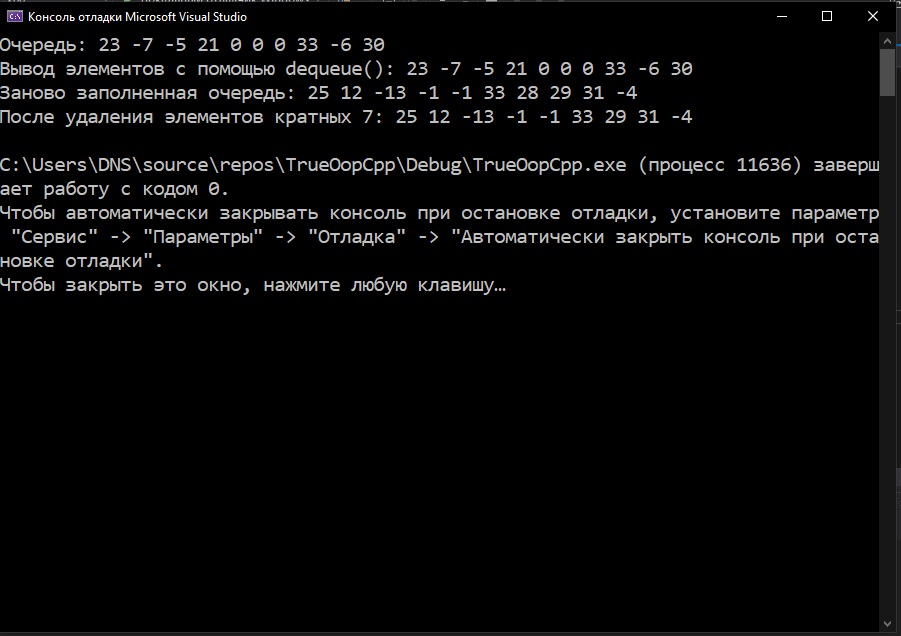
Оценка (балл):\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020г.

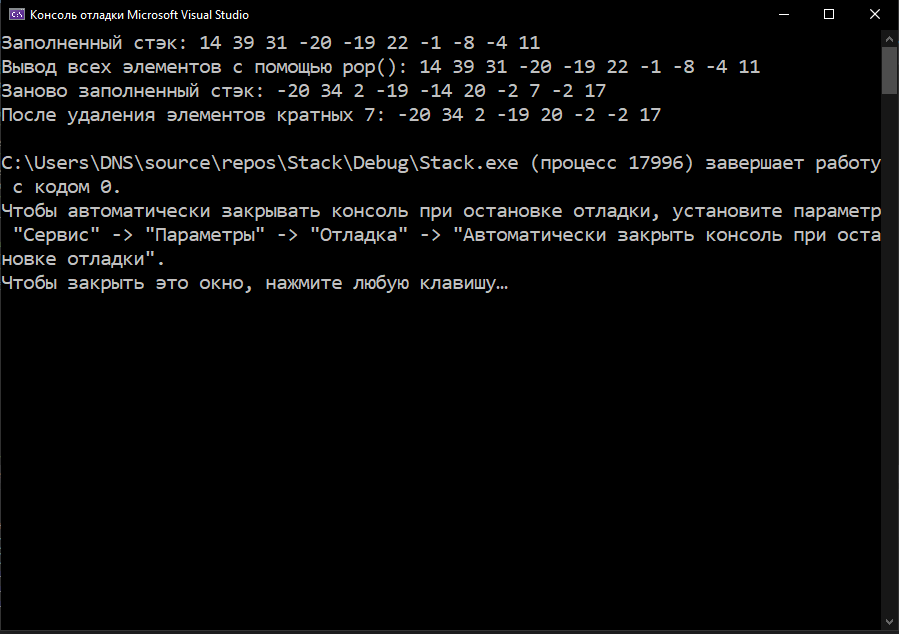
Задание:

Результат работы программ:

Очередь



Стэк



Заполнение элементов стека и очереди происходит рандомно, поэтому может случиться так, что не окажется чисел, которые кратны 7-ми. Для решения данной проблемы нужно просто перезапустить программу.

Реализация:

Очередь:

#include <iostream>

#include <ctime>

using namespace std;

/\*

Это шаблон,при ининиализации можно пихать любые типы,

но это больше для виду, чтобы эта очередь была похожа на настоящую очередь

\*/

template<class T>

class Queue { // класс очередь

private: // приватные поля

class Node { // класс узла

public: // публичные поля

T data; // значение в узле

Node\* next; // указатель на следующий элемент

};

Node\* head; // "голова" очереди

Node\* tail; // "хвост очереди"

int size; // размер

public: // публичные поля

// проверка на пустоту

bool is\_empty()

{ return head == nullptr;}

// конструктор без параметров

Queue() {

// присваиваем значения

head = nullptr;

tail = nullptr;

size = 0;

}

// геттер

int get\_size() {

// возвращает размер очереди

return size;

}

// функция добавления в очередь

void enqueue(T x) {

// временный узел

Node \*temp = new Node;

// присваиваем значение

temp->data = x;

temp->next = nullptr;

// если в очередь пустая

if (is\_empty()) {

// добавляем

head = temp;

tail = temp;

// увеличиваем размер

size++;

}

// если очередь не пустая

else {

// добавляем в конец

tail->next = temp;

tail = temp;

// увеличиваем размер

size++;

}

}

// удаление из очереди

T dequeue() {

// запоминаем значение

T data = head->data;

// запоминаем первый элемент

Node\* temp = head;

head = head->next;

// удаляем

delete temp;

// уменьшаем размер

size--;

// возвращаем значение удаленного узла

return data;

}

// вывод в консоль

void print() {

// узел

Node\* temp;

// если очередь не пуста

if (!is\_empty()) {

// идем в цикле

for (temp = head; temp != nullptr; temp = temp->next) {

// и выводим

cout << temp->data << " ";

}

cout << endl;

}

// иначе выводим,что очередь пуста

else {

cout << endl << "Очередь пуста!";

}

}

// удаление по значению

void delete\_by\_condition(T data) {

// двойной указатель на первый узел

Node\*\* temp = &head;

// указатель на текущий узел

Node\* current = head;

// пока текущий узел != null

while (current) {

// если значение в узле делится на значение по условию

if (current->data % data == 0) {

// меняем указатель

\*temp = current->next;

// удаляем ноду

delete current;

// уменьшаем размер

size--;

// присваиваем новый указатель

current = \*temp;

}

// иначе

else {

// просто идем дальше

temp = &current->next;

current = current->next;

}

}

}

};

int main() {

// для рандома

srand(time(NULL));

// русский язык

setlocale(LC\_ALL, "rus");

// созлдаем

Queue<int> queue;

// заполняем

for (int i = 0; i < 10; i++) {

int x = rand() % 60 + (-20);

queue.enqueue(x);

}

cout << "Очередь: ";

// выводим

queue.print();

cout << "Вывод элементов с помощью dequeue(): ";

while (queue.get\_size() > 0) {

cout << queue.dequeue() << " ";

}

cout << endl;

// заполнение очереди заново,т.к. все элементы были удалены

for (int i = 0; i < 10; i++) {

int x = rand() % 60 + (-20);

queue.enqueue(x);

}

cout << "Заново заполненная очередь: ";

queue.print();

cout << "После удаления элементов кратных 7: ";

// удаляем

queue.delete\_by\_condition(7);

// выводим после удаления

queue.print();

return 0;

}

Стэк

#include <iostream>

#include <ctime>

using namespace std;

/\*

Это шаблон,при ининиализации можно пихать любые типы,

но это больше для виду, чтобы эта очередь была похожа на настоящую очередь

\*/

template<class T>

class Stack { // класс стэк

private: // приватные поля

class Node { // класс узел

public: // его публичные поля

T data; // значение

Node\* next; // указатель на следующий узел

};

int size; // размер

Node\* top; // указатель на "топ"

public:

// конструктор

Stack() {

// инициализируем переменные

top = nullptr;

size = 0;

}

//проверка на пустоту

bool is\_empty() {

return size == 0;

}

// геттер

int get\_size() {

return size;

}

// добавление в стек

void push(T value) {

// временный узел

Node\* temp = new Node;

// присваиваем значения

temp->data = value;

// добавляем в начало

temp->next = top;

top = temp;

// уеличиваем размер

size++;

}

// удаление из стека

T pop() {

// если стек не пуст

if (top != nullptr) {

// значение

int value = top->data;

// удаляем узел

Node\* temp = top;

top = temp->next;

delete temp;

// уменьшаем размер

size--;

// возвращаем значение удаленного узла

return value;

}

// если стек пуст выводим сообщение

else {

cout << "Стэк пуст!" << endl;

return 0;

}

}

// вывод в консоль

void print() {

// узел

Node\* temp;

// если очередь не пуста

if (!is\_empty()) {

// идем в цикле

for (temp = top; temp != nullptr; temp = temp->next) {

// и выводим

cout << temp->data << " ";

}

cout << endl;

}

// иначе выводим,что очередь пуста

else {

cout << endl << "Очередь пуста!";

}

}

// удаление по значению

void delete\_by\_condition(T data) {

// двойной указатель на первый узел

Node\*\* temp = &top;

// указатель на текущий узел

Node\* current = top;

// пока текущий узел != null

while (current) {

// если значение в узле делится на значение по условию

if (current->data % data == 0) {

// меняем указатель

\*temp = current->next;

// удаляем ноду

delete current;

// уменьшаем размер

size--;

// присваиваем новый указатель

current = \*temp;

}

// иначе

else {

// просто идем дальше

temp = &current->next;

current = current->next;

}

}

}

};

int main() {

// для рандома

srand(time(NULL));

// русский язык

setlocale(LC\_ALL, "rus");

Stack<int> stack;

for (int i = 0; i < 10; i++) {

int x = rand() % 60 + (-20);

stack.push(x);

}

cout << "Заполненный стэк: ";

stack.print();

cout << "Вывод всех элементов с помощью pop(): ";

while (stack.get\_size() > 0) {

cout << stack.pop() << " ";

}

cout << endl;

// заполнение стэка заново,т.к. все элементы были удалены

for (int i = 0; i < 10; i++) {

int x = rand() % 60 + (-20);

stack.push(x);

}

cout << "Заново заполненный стэк: ";

stack.print();

cout << "После удаления элементов кратных 7: ";

stack.delete\_by\_condition(7);

stack.print();

return 0;

}