МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| Кафедра математического и программного обеспечения |
| Структуры и алгоритмы обработки данных |
|  |

ЛАБОРАТОРНАЯ РАБОТА №1

|  |  |
| --- | --- |
| Исполнитель: | студент  группы 1ПИб-02-3оп-23  Богданов  Ренат Алексеевич |
| Руководитель: | Журавлёва Юлия Михайловна |
| Оценка: |  |
| Подпись: |  |

2024 год

Задача

Выполнить задания, используя для представления очередей и стеков:

а) массивы;

б) динамические списки.

19. Даны очередь и стек целых чисел. Из элементов очереди меньших максимального элемента стека сформировать новый стек.

Требования к программам

Количество элементов исходных линейных списков заранее не определено и задается случайным образом. При дальнейшей обработке считается, что количество элементов списка не известно, т.е. обработка производится, пока не достигнут конец списка.

Программа должна сформировать исходные линейные списки, вывести их содержимое на экран (при этом данные из списков не должны быть потеряны), произвести обработку и вывести содержимое итогового списка на экран.

Код программы:

// 19. Даны очеpедь и стек целых чисел. Из элементов очереди меньших максимального элемента стека сфоpмиpовать новый стек.

// Стек и очередь массивом

#include <chrono>

#include <iostream>

// Реализация стека массивом

struct Stack1 {

int top;

int\* data;

};

void initStack(Stack1 &st, int capacity) {

st.data = new int[capacity];

st.top = -1;

}

void push(Stack1 &st, int value) {

st.data[++st.top] = value;

}

int pop(Stack1 &st) {

return st.data[st.top--];

}

void nullStack(Stack1 &st) {

st.top = -1;

}

bool empty(Stack1 &st) {

return st.top == -1;

}

// -------------------------------------------

// Реализация очереди массивом

struct Queue1 {

int head, tail, size;

int\* data;

};

void nullQueue(Queue1& q) {

q.head = 0; q.tail = q.size - 1;

}

void initQueue(Queue1& q, int capacity) {

q.size = capacity+1;

q.data = new int[q.size];

nullQueue(q);

}

int next(Queue1& q, int n) {

return (n + 1) % q.size;

}

bool empty(Queue1& q) {

return next(q, q.tail) == q.head;

}

void add(Queue1& q, int value) {

if (next(q, next(q, q.tail)) == q.head)

std::cout << "Queue owerflow" << std::endl;

else {

q.tail = next(q, q.tail);

q.data[q.tail] = value;

}

}

int del(Queue1& q) {

if (empty(q)) {

std::cout << "Queue is empty" << std::endl;

return 0;

}

else {

int d = q.data[q.head];

q.head = next(q, q.head);

return d;

}

}

// -------------------------------------------

// Реализация стека динамическим списком

struct Node\_stack {

int data;

Node\_stack\* next;

};

void initStack(Node\_stack\*& top) {

top = NULL;

}

void push(Node\_stack\*& top, int value) {

Node\_stack\* tmp = new Node\_stack;

tmp->next = top;

top = tmp;

top->data = value;

}

int pop(Node\_stack\*& top) {

Node\_stack\* tmp = top;

int d = top->data;

top = top->next;

delete(tmp);

return d;

}

bool empty(Node\_stack\*& top) {

return top == NULL;

}

void nullStack(Node\_stack\*& top) {

Node\_stack\* tmp;

while (!empty(top)) {

tmp = top;

top = top->next;

delete(tmp);

}

}

// -------------------------------------------

// Реализация очереди динамическим списком

class Queue2 {

private:

struct Node {

int data;

Node\* next;

};

Node\* head, \* tail;

public:

Queue2() {

head = NULL;

tail = NULL;

}

bool empty() {

return head == NULL;

}

void add(int value) {

if (empty()) {

head = new Node;

head->data = value;

head->next = NULL;

tail = head;

}

else {

tail->next = new Node;

tail = tail->next;

tail->data = value;

tail->next = NULL;

}

}

int del() {

if (empty()) {

std::cout << "Queue is empty" << std::endl;

return 0;

}

else {

int d = head->data;

Node\* tmp = head;

head = head->next;

delete(tmp);

return d;

}

}

void nullQueue() {

Node\* tmp;

while (!empty()) {

tmp = head;

head = head->next;

delete(tmp);

}

}

};

// -------------------------------------------

// Программа на массивах

void massive() {

Stack1 stack;

Queue1 queue;

Stack1 result;

std::cout << "~Massive" << std::endl;

// Заполнение стека

std::cout << "Stack:" << std::endl;

int stackLen = rand() % 20 + 1;

initStack(stack, stackLen);

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

int newEl = rand() % 10;

push(stack, newEl);

std::cout << newEl << " ";

}

std::cout << std::endl;

// Поиск максимального элемента и вывод содержимого

std::cout << "Stack (pop):" << std::endl;

int maxEl = 0;

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

int currentEl = pop(stack);

if (maxEl < currentEl) maxEl = currentEl;

std::cout << currentEl << " ";

}

std::cout << std::endl;

std::cout << "Max in stack: " << maxEl << std::endl;

// Заполнение очереди

int queueLen = rand() % 20 + 1;

std::cout << "Queue len: " << queueLen << std::endl;

std::cout << "Queue: " << std::endl;

initQueue(queue, queueLen);

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

int newQEl = rand() % 10;

add(queue, newQEl);

std::cout << newQEl << " ";

}

std::cout << std::endl;

//Заполнение стека с результатом

initStack(result, queueLen);

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

int curQueue = del(queue);

if (curQueue < maxEl) {

push(result, curQueue);

}

}

//Вывод результата

std::cout << "Result (pop): " << std::endl;

while (!empty(result)) {

std::cout << pop(result) << " ";

}

std::cout << std::endl << std::endl;

};

// Программа на динамических списках

void dynamic\_list() {

std::cout << "~Dynamic list" << std::endl;

Node\_stack\* stack;

Queue2 queue;

Node\_stack\* result;

// Заполнение стека

initStack(stack);

std::cout << "Stack:" << std::endl;

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

int newEl = rand() % 10;

push(stack, newEl);

std::cout << newEl << " ";

}

std::cout << empty(stack) << std::endl;

// Поиск максимального элемента и вывод содержимого

std::cout << "Stack (pop):" << std::endl;

int maxEl = 0;

while (!empty(stack)) {

int currentEl = pop(stack);

if (maxEl < currentEl) maxEl = currentEl;

std::cout << currentEl << " ";

}

std::cout << std::endl;

std::cout << "Max in stack: " << maxEl << std::endl;

// Заполнение очереди

int queueLen = rand() % 20 + 1;

std::cout << "Queue len: " << queueLen << std::endl;

std::cout << "Queue: " << std::endl;

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

int newQEl = rand() % 10;

queue.add(newQEl);

std::cout << newQEl << " ";

}

std::cout << std::endl;

//Заполнение стека с результатом

initStack(result);

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

int curQueue = queue.del();

if (curQueue < maxEl) {

push(result, curQueue);

}

}

//Вывод результата

std::cout << "Result (pop): " << std::endl;

while (!empty(result)) {

std::cout << pop(result) << " ";

}

std::cout << std::endl << std::endl;

}

// Основная программа

int main() {

srand(time(0));

massive();

std::cout << std::endl << std::endl << std::endl << std::endl;

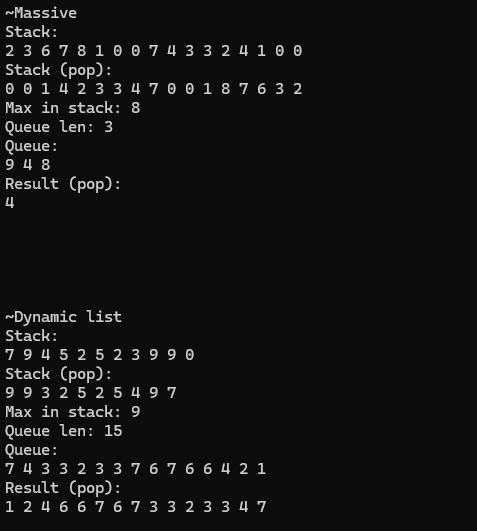
dynamic\_list();

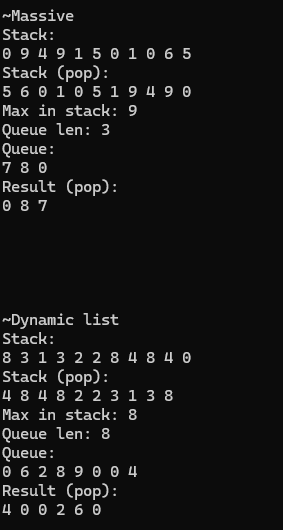
system("pause");

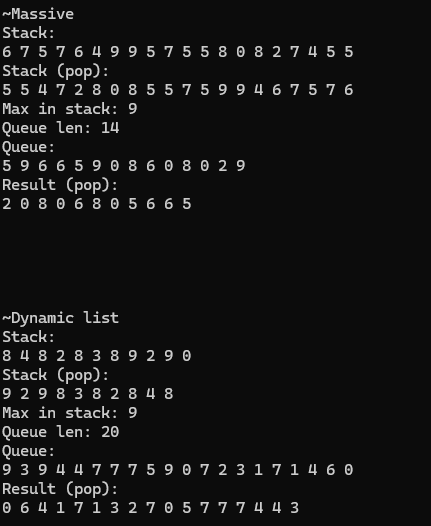
return 0;

}

Примеры работы программы







Вывод

В ходе выполнения лабораторной работы были освоены работа со структурами данных типа стек и очередь.