**Лабоpатоpная pабота №1**

**Линейные списки**

Выполнить задания, используя для пpедставления очеpедей и стеков:

а) массивы; б) динамические списки.

Требования к программам:

1. 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.

2. 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.

Вариант 16

16. Даны стек и очеpедь целых чисел. Из элементов очереди, на котоpые

не делится сумма элементов стека сфоpмиpовать новый стек.

Алгоритм (а):

1) Описывается структура Stack, где объявляются переменные:

top - индекс верхнего элемента стека, data - указатель на элемент массива.

Описывается структура Queue, где объявляются переменные: head - поле, где хранится позиция первого элемента, tail - поле, где хранится позиция последнего, size - размер массива, data - указатель на первый элемент массива

2) Описываются функции стека и очереди.

Программы (а):

#include <iostream>

#include <string>

#include <Time.h>

using namespace std;

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;

}

void showStack(Stack& st, int top) {

for (int i = top; i > -1; i--)

cout << i << " " << st.data[i] << endl;

}

struct Queue {

int head, tail, size;

int\* data;

};

void nullQueue(Queue& q) {

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

}

void InitQueue(Queue& q, int capacity) {

q.size = capacity;

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

nullQueue(q);

}

int next(Queue& q, int n) {

return (n + 1) % q.size;

}

bool empty(Queue& q) {

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

}

void add(Queue& q, int value) {

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

cout << "Queue is overflow" << endl;

}

else

{

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

q.data[q.tail] = value;

}

}

void showQueue(Queue& q) {

if (empty(q)) { cout << "Queue is empty" << endl; }

else {

int i = q.head;

while (i != q.tail + 1) {

cout << q.data[i] << " ";

i = next(q, i);

} cout << endl;

}

}

int main() {

srand(time(0));

int value1, value2, tmp, a, summ=0, k = 0;

double n1 = rand() % 10 + 1;

double n2 = rand() % 10 + 1;

Stack1 st1, st2 ;

Queue q;

InitStack(st1, n1);

InitStack(st2, n2);

for (int i = 0; i < n1; i++) {

value1 = rand() % 10 + 1;

push(st1, value1);

}

for (int i = 0; i < n2; i++) {

value2 = rand() % 15 + 1;

push(st2, value2);

}

cout << "Stack 1: ";

tmp = st1.top;

while (!empty(st1)) {

a = pop(st1);

cout << a << " ";

}

st1.top = tmp;

cout << endl;

cout << "Stack 2: ";

tmp = st2.top;

while (!empty(st2)) {

a = pop(st2);

cout << a << " ";

}

st2.top = tmp;

cout << endl;

while (!empty(st2)) {

a = pop(st2);

summ += a;

}

st2.top = tmp;

cout << "Sum 2nd stack = " << summ;

cout << endl;

InitQueue(q, n1);

tmp = st1.top;

while (!empty(st1)) {

a = pop(st1);

if (summ % a != 00) {

add(q, a);

k++;

}

}

if (k == 0) cout << "there are no valid values";

else {

cout << "Queue: ";

st1.top = tmp;

while (!empty(q)) {

a = del(q);

cout << a << " ";

}

}

cout << endl;

system("pause");

return 0;

}