ФГБОУ ВО “Чувашский государственный университет им.

И.Н. Ульянова”

Кафедра вычислительной техники

Лабораторная работа №1

Стеки. Очереди. Деки.

Вариант 9.

Выполнил: студент гр. Ивт-41-22

Иванов В.С.

Проверил: кандидат технических наук

Павлов Л.А.

***Цель работы*** – изучить основные способы реализации стеков, очередей и деков, выполняемые над ними операции; получить практические навыки программирования задач с использованием стеков, очередей, деков.

Подготовка к работе

Реализовать стек и очередь с использованием последовательного (массив) и связанного (динамические списковые структуры) распределений элементов. Для каждого способа представления написать подпрограммы, реализующие операции над стеками и очередями:

*1)Empty*(*S*) – проверка на наличие элементов (проверка пустоты стека):

Массив

**if** t=0

**then** {True}

**else** {False}

Односвязный список

**if** t=NULL

**then**{True}

**else**{False}

2)Push(S, x) – в вершину стека помещает элемент x:

Массив

t := t+1

**if** t > m

**then** {переполнение}

**else** S(t) := x

Односвязный список

new(l)

l.info := x

l.next := t

t := l

*3)Pop*(*S*) – удаляет элемент из вершины стека *S* и возвращает его значение (не определена для пустого стека);

Массив

**if** t = 0

**then** {Стек пуст}

**else** {x = S(t)

t = t-1}

Односвязный список

**if** t = NULL

**then**{стек пуст}

**else**{x := t.info

l := t

t := t.next

dispose(l)}

4) *StackTop*(*S*) – возвращает значение элемента в вершине стека *S* без его удаления (не определена для пустого стека):

Массив

**if** t=0

**then** {Стек пуст}

**else** {x = S(t)}

Односвязный список

**if** t = NULL

**then**{стек пуст}

**else**{x := t.info}

2. Реализовать очередь с использованием последовательного (массив) и связного (динамические списковые структуры) распределений элементов. Для каждого способа представления разработать алгоритмы, реализующие выполнение следующих операций над очередями:

1) *EmptyQueue*(*Q*) – проверка на наличие элементов (проверка пустоты очереди):

Массив

**if** r=f

**then** {True}

**else** {False}

Односвязный список

**if** f=r

**then** {True}

**else** {False}

2) *InsQueue*(*Q*, *x*) – помещает элемент *x* в конец очереди *Q:*

Массив

r = (r+1)**mod** m

**if** r = f

**then** {переполнение}

**else** Q(r)= x

Односвязный список

new(r.next)

r = r.next

r.info = x

r.next = NULL

3) *DelQueue*(*Q*) – удаляет элемент из начала очереди *Q* и возвращает его значение (не определена для пустой очереди):

Массив

**if** r = f

**then**{очередь пуста}

**else** {f := (f+1)**mod m**

x := Q(f)

Односвязный список

**if** f=r

**then**// очередь пуста

**else** x := f.next.info

l := f

f := f.next

dispose(l)

3. Реализовать дек с использованием последовательного (массив) и связного (динамические списковые структуры) распределений элементов. Для каждого способа представления разработать алгоритмы, реализующие выполнение следующих операций над деками:

1) *Empty*(*D*) – проверка на наличие элементов (проверка пустоты дека):

**if** h=0 and t=m

**then**{True}

**else**{False}

2) *PushBack*(*D*, *x*) – помещает элемент *x* в конец дека *D*:

**if** h=t

**then**{Дек переполнен}

**else**{t := t-1

d(t) := x}

3) *PushFront*(*D*, *x*) – помещает элемент *x* в начало дека *D*:

**if** h = t

**then**{Дек переполнен}

**else**{

h := h-1

d(h) := x }

4) *PopBack*(*D*) – удаляет элемент из конца дека *D* и возвращает его значение (не определена для пустого дека):

**if** h = 0

**then** {Дек пуст}

**else** {x := d(t)

t := t-1}

5) *PopFront* (*D*) – удаляет элемент из начала дека *D* и возвращает его значение (не определена для пустого дека):