1. Постановка задачи

Разработать программу для работы с информационно-логическими структурами – стеком, очередью и деком. Задание: создать стек, очередь и дек, содержащие целые числа в диапазоне от -30 до +50. Из стека и очереди удалить все элементы из диапазона от -10 до +30, из дека удалить элементы из диапазона от -10 до +30, если они при этом находятся на одинаковом расстоянии от краёв дека.

Структуры реализовать на основе списка и массива.

2. Входные и выходные данные

2.1 Входные данные

Основная программа (на основе списка):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |
| 2 | qu | tqueue | Пр. пер. | Очередь | - |
| 3,4 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Основная программа (на основе массива):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |
| 2 | qu | tqueue | Пр. пер. | Очередь | - |
| 3 | de | tdeck | Пр. пер. | Дек | - |

Unit со Стеком:

Функция IsEmptyStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура AddStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Функция IsOneElemStack (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Функция GetInfoBeginStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура TopToTopStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Функция IsEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура AddToEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Функция GetInfoEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура AddMiddleStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | o | tinfo | Пр. пер. | После какого эл-та вставить | ХХ |
| 2 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Процедура DeleteMiddleStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Unit с Очередью:

Функция IsEmptyQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Функция IsOneElemQueue (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура AddQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Функция TakeInfoBeginQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура AddtoBeginQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Процедура AddMiddleQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | o | tinfo | Пр. пер. | После какого эл-та вставить | ХХ |
| 2 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Процедура DeleteMiddleQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Функция TakeInfoEndQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура TopToBottomQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Unit с Деком:

Функция IsEmptyDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Функция IsEmptyDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура AddBeginDeck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Процедура AddEndDeck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | n | tinfo | Пр. пер. | Значение эл-та | ХХ |

Функция GetInfoBeginDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Функция GetInfoBeginDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Функция GetInfoEndDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Функция GetInfoEndDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура TopToTopDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура TopToTopDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура BottomToBottomDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура BottomToBottomDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Функция IsOneElemDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Функция IsOneElemDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура BottomToTopDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура BottomToTopDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура TopToBottomDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура TopToBottomDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

2.2 Выходные данные

Основная программа (на основе списка):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек после удаления | - |
| 2 | qu | tqueue | Пр. пер. | Очередь после удаления | - |
| 3,4 | de1,de2 | tdeck | Пр. пер. | Дек после удаления | - |

Основная программа (на основе массива):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек после удаления | - |
| 2 | qu | tqueue | Пр. пер. | Очередь после удаления | - |
| 3 | de | tdeck | Пр. пер. | Дек после удаления | - |

Unit со Стеком:

Процедура InitStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Функция IsEmptyStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsEmptyStack | Лог. | Пр. пер. | Проверка на пустоту | - |

Функция IsOneElemStack (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsOneElemStack | Лог. | Пр. пер. | Проверка на единственность элемента | - |

Процедура AddStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура DeleteElemStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Функция GetInfoBeginStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | GetInfoBeginStack | tinfo | Пр. пер. | Значение  1-го эл-та | ХХ |

Процедура TopToTopStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | buf | tstack | Пр. пер. | Буферный стек | - |

Процедура DeleteListStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура QuestStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | fl | Лог. | Пр. пер. | Флажок | - |
| 2 | st | tstack | Пр. пер. | Стек | - |

Процедура AddToEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура DeleteEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Функция GetInfoEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | GetInfoEndStack | tinfo | Пр. пер. | Значение  последнего эл-та | ХХ |

Функция IsEndStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsEndStack | Лог. | Пр. пер. | Проверка на окончание | - |

Процедура AddMiddleStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Процедура DeleteMiddleStack

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | st | tstack | Пр. пер. | Стек | - |

Unit c Очередью:

Функция IsEmptyQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsEmptyQueue | Лог. | Пр. пер. | Проверка на пустоту | - |

Процедура AddQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура DeleteElemQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Функция IsOneElemQueue (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsOneElemQueue | Лог. | Пр. пер. | Проверка на единственность элемента | - |

Процедура InitQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура DeleteListQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Функция TakeInfoBeginQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | TakeInfoBeginQueue | tinfo | Пр. пер. | Значение  1-го эл-та | ХХ |

Процедура TopToBottomQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | buf | tqueue | Пр. пер. | Буферная очередь | - |

Процедура QuestQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | fl | Лог. | Пр. пер. | Флаг | - |
| 2 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура AddtoBeginQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура AddMiddleQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура DeleteEndQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Процедура DeleteMiddleQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | qu | tqueue | Пр. пер. | Очередь | - |

Функция TakeInfoEndQueue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | TakeInfoEndQueue | tinfo | Пр. пер. | Значение  последнего эл-та | ХХ |

Unit с Деком:

Функция IsEmptyDeck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsEmptyDeck | Лог. | Пр. пер. | Проверка на пустоту | - |

Процедура AddBeginDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура AddBeginDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура AddEndDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура AddEndDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура DeleteElemBeginDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура DeleteElemBeginDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура DeleteElemEndDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура DeleteElemEndDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Процедура DeleteListDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура DeleteListDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Функция GetInfoBeginDeck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | GetInfoBeginDeck | tinfo | Пр. пер. | Значение  1-го эл-та | ХХ |

Функция GetInfoEndDeck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | GetInfoEndDeck | tinfo | Пр. пер. | Значение  последнего эл-та | ХХ |

Процедура TopToTopDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | buf1,buf2 | tdeck | Пр. пер. | Буферный дек | - |

Процедура TopToTopDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | buf | tdeck | Пр. пер. | Буферный дек | - |

Процедура BottomToBottomDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | buf1,buf2 | tdeck | Пр. пер. | Буферный дек | - |

Процедура BottomToBottomDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | buf | tdeck | Пр. пер. | Буферный дек | - |

Процедура InitDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура InitDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | de | tdeck | Пр. пер. | Дек | - |

Функция IsOneElemDeck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | IsOneElemDeck | Лог. | Пр. пер. | Проверка на единственный эл-т | - |

Процедура QuestDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | fl | Лог. | Пр. пер. | Флаг | - |
| 2,3 | de1,de2 | tdeck | Пр. пер. | Дек | - |

Процедура QuestDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | fl | Лог. | Пр. пер. | Флаг | - |
| 2 | de | tdeck | Пр. пер. | Дек | - |

Процедура BottomToTopDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | buf1,buf2 | tdeck | Пр. пер. | Буферный дек | - |

Процедура BottomToTopDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | buf | tdeck | Пр. пер. | Буферный дек | - |

Процедура TopToBottomDeck (на основе списка)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1,2 | buf1,buf2 | tdeck | Пр. пер. | Буферный дек | - |

Процедура TopToBottomDeck (на основе массива)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Имя | Тип | Структура | Значение | Формат |
| 1 | buf | tdeck | Пр. пер. | Буферный дек | - |

3. Аномалии

Всевозможные аномалии с текстовыми файлами: Невозможно открыть файл для чтения, невозможно открыть файл для записи, некорректные данные в файле, значения в текстовом файле не попадают в заданный диапазон – такие значения должны отбрасываться, добавляемое значение не попадает в заданный диапазон – такие значения не должны добавляться.

4. Метод

Берём элемент из стека. Если он не удовлетворяет условию, кладём его в другой стек. Таким образом, выбираем все элементы исходного стека. Когда исходный стек станет пуст, перекладываем элементы из дополнительного стека обратно в исходный.

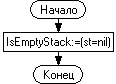
С очередью поступаем аналогично – перемещаем элементы, не удовлетворяющие условию, из исходной в дополнительную очередь, а потом обратно.

Для обработки дека можно использовать либо одну, либо две дополнительные структуры. Берём по элементу с каждой стороны дека. Если хотя бы один из них не удовлетворяет условию, кладём пару элементов либо в один дополнительный дек с разных сторон, либо в разные дополнительные деки. После освобождения исходного дека, перекладываем элементы обратно.

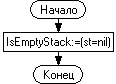
5. Алгоритм

Стек (на основе списка)

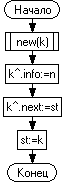
Процедура InitStack



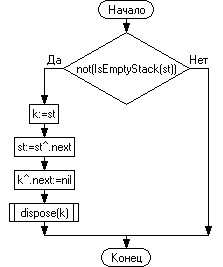
Функция IsEmptyStack



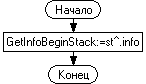
Процедура AddStack



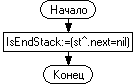
Процедура DeleteElemStack



Функция GetInfoBeginStack

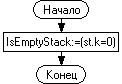


Функция IsEndStack

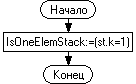


Стек (на основе массива)

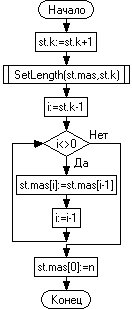
Функция IsEmptyStack



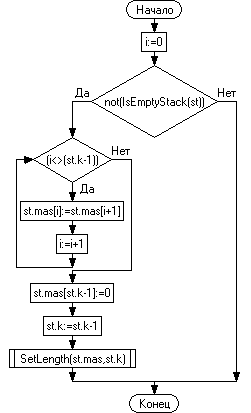
Функция IsEndStack



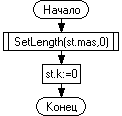
Процедура AddStack



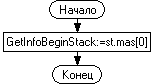
Процедура DeleteElemStack



Процедура InitStack

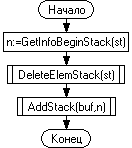


Функция GetInfoBeginStack

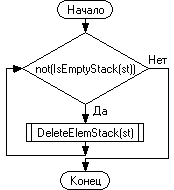


Стек (общие)

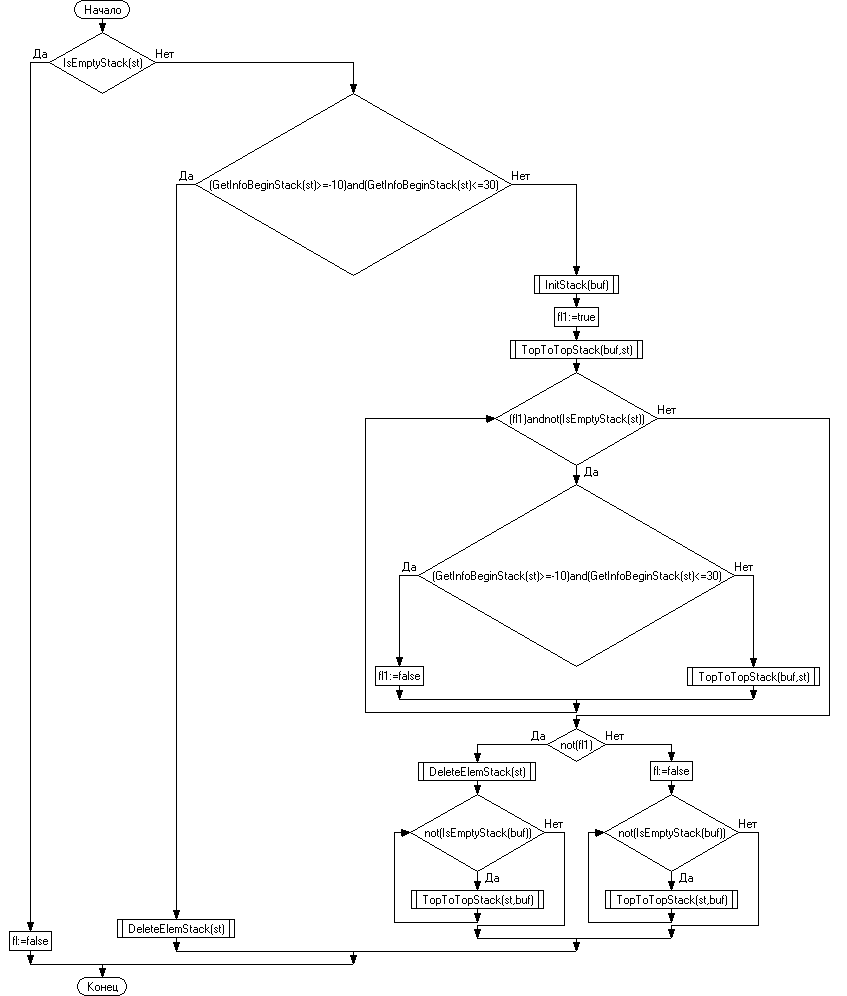
Процедура TopToTopStack



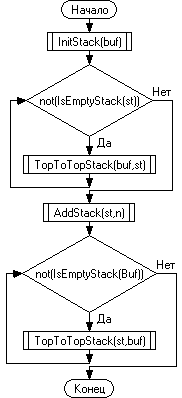
Процедура DeleteListStack



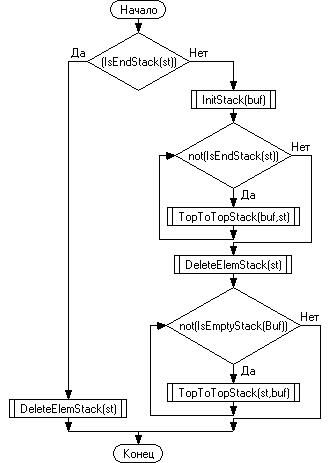
Процедура QuestStack



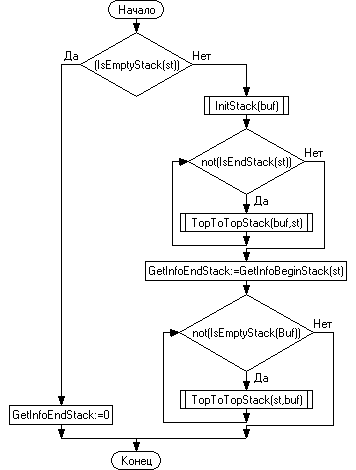
Процедура AddToEndStack



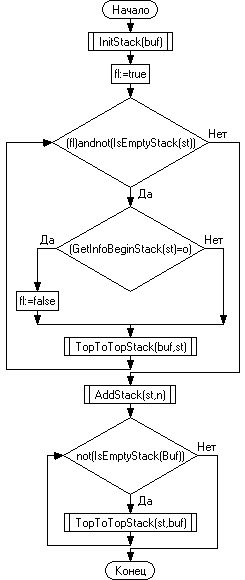
Процедура DeleteEndStack



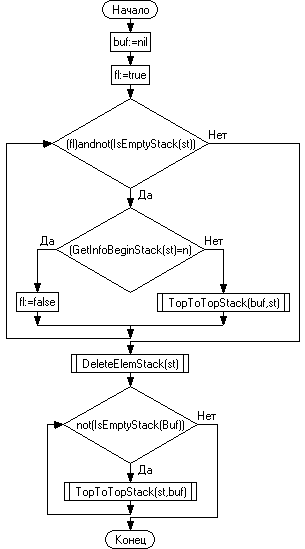
Функция GetInfoEndStack



Процедура AddMiddleStack

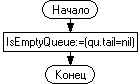


Процедура DeleteMiddleStack

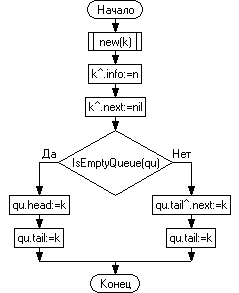


Очередь (на основе списка)

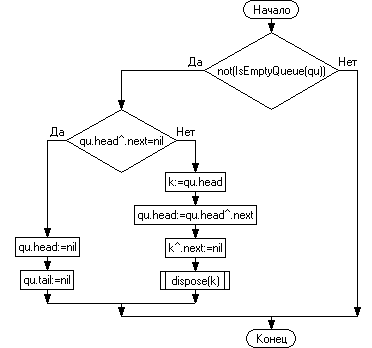
Функция IsEmptyQueue



Процедура AddQueue



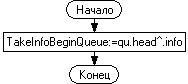
Процедура DeleteElemQueue



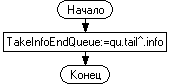
Процедура InitQueue



Функция TakeInfoBeginQueue

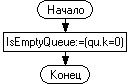


Функция TakeInfoEndQueue

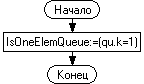


Очередь (на основе массива)

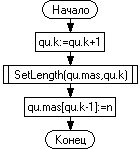
Функция IsEmptyQueue



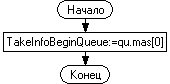
Функция IsOneElemQueue



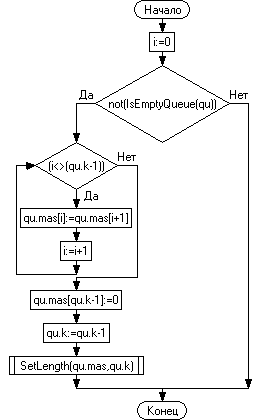
Процедура AddQueue



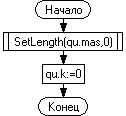
Функция TakeInfoBeginQueue



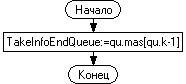
Процедура DeleteElemQueue



Процедура InitQueue

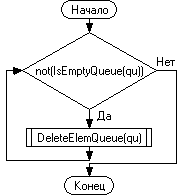


Функция TakeInfoEndQueue

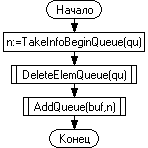


Очередь (общие)

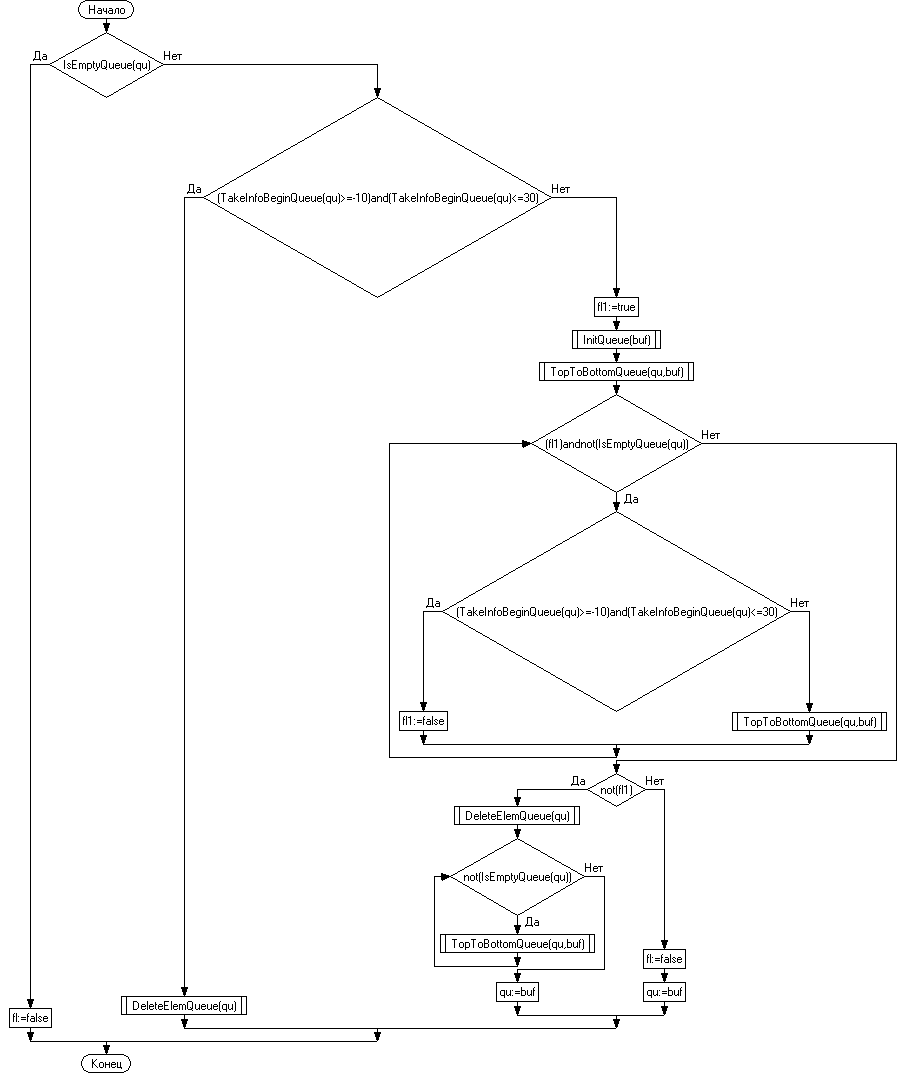
Процедура DeleteListQueue



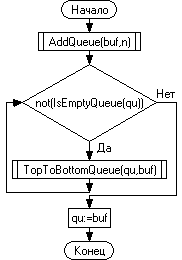
Процедура TopToBottomQueue



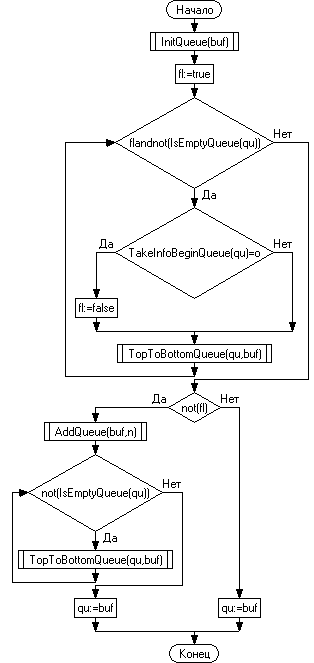
Процедура QuestQueue



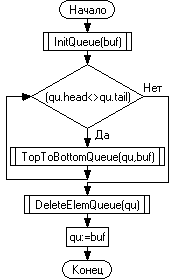
Процедура AddtoBeginQueue



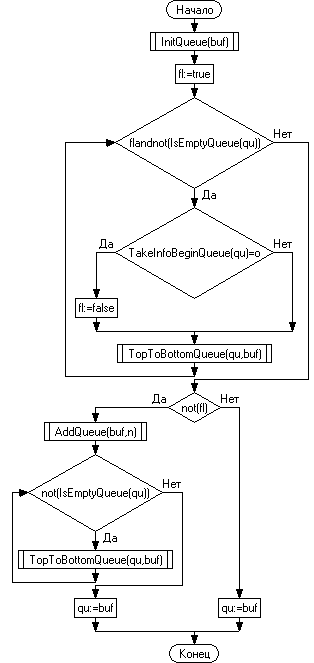
Процедура AddMiddleQueue



Процедура DeleteEndQueue

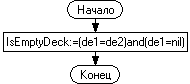


Процедура DeleteMiddleQueue

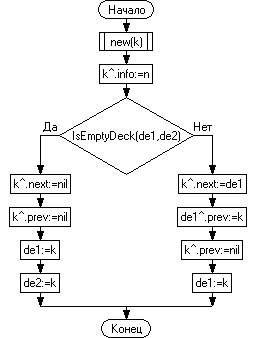


Дек (на основе списка)

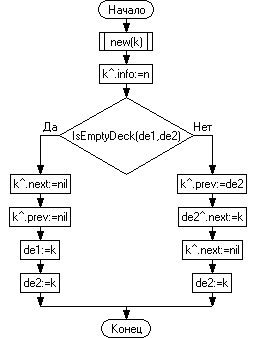
Функция IsEmptyDeck



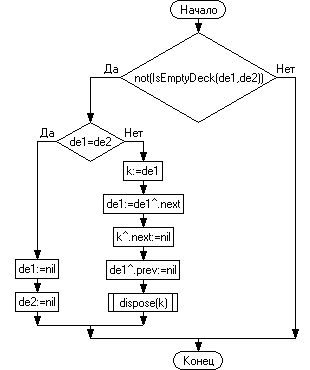
Процедура AddBeginDeck



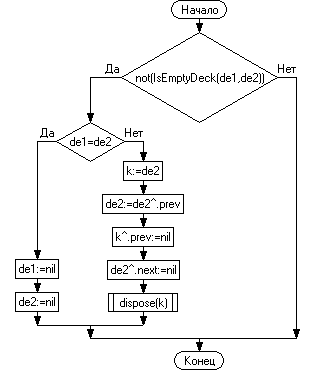
Процедура AddEndDeck



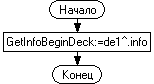
Процедура DeleteElemBeginDeck



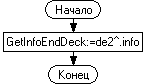
Процедура DeleteElemEndDeck



Функция GetInfoBeginDeck



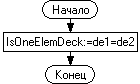
Функция GetInfoEndDeck



Процедура InitDeck

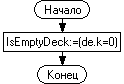


Функция IsOneElemDeck

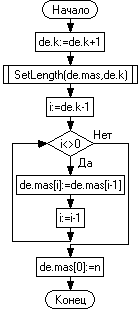


Дек (на основе массива)

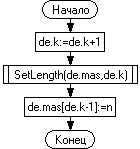
Функция IsEmptyDeck



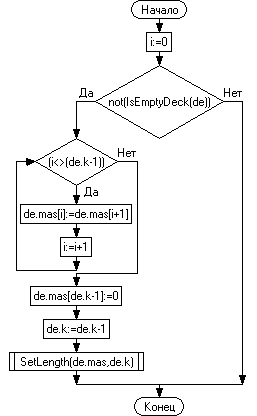
Процедура AddBeginDeck



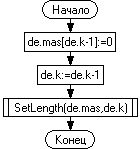
Процедура AddEndDeck



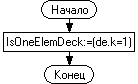
Процедура DelElemBeginDeck



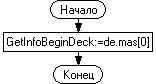
Процедура DelElemEndDeck



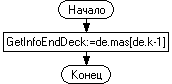
Функция IsOneElemDeck



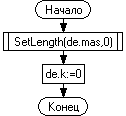
Функция GetInfoBeginDeck



Функция GetInfoEndDeck

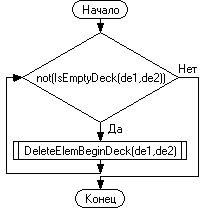


Процедура InitDeck

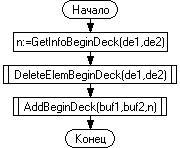


Дек (общие)

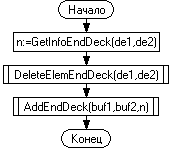
Процедура DeleteListDeck



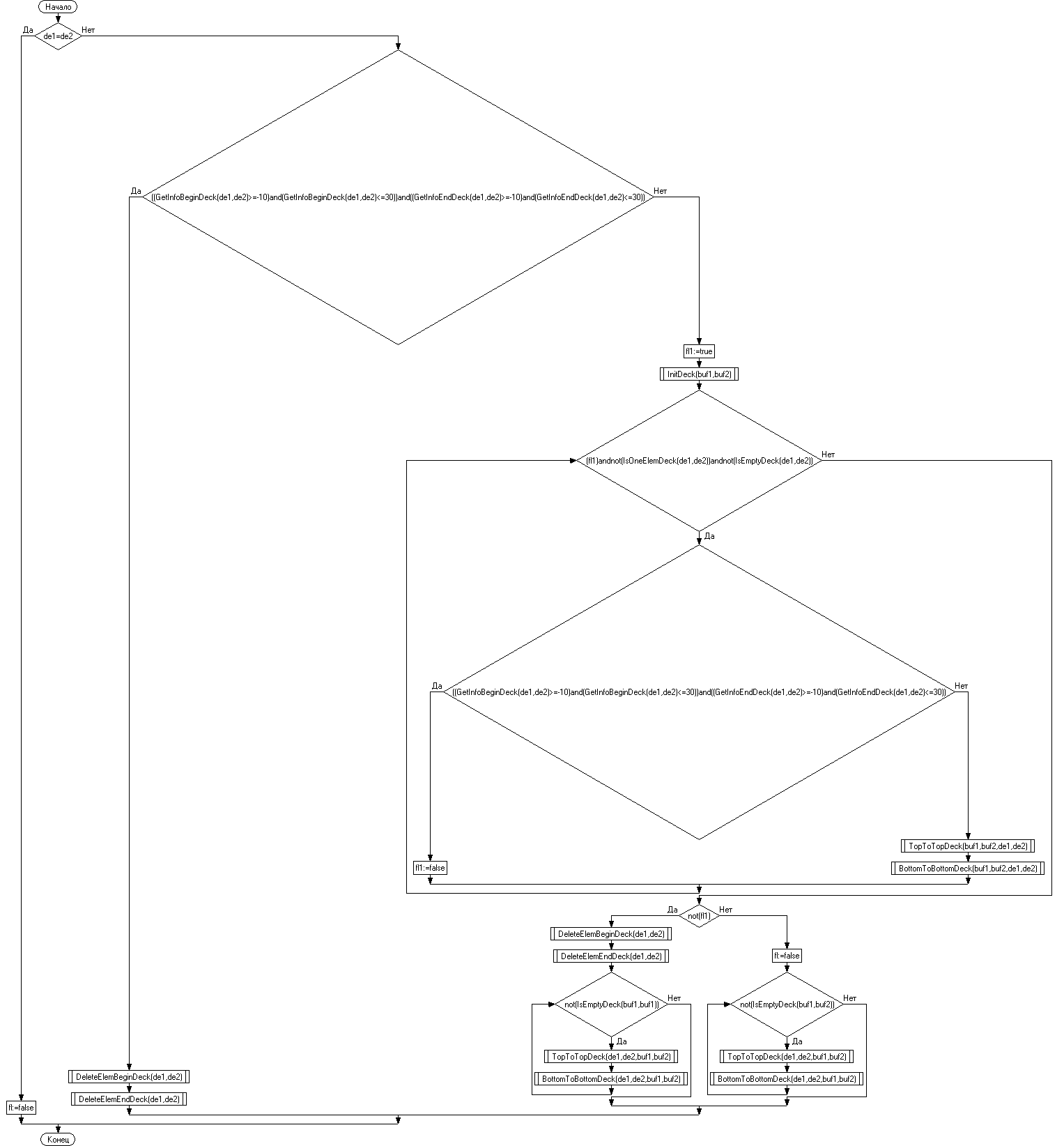
Процедура TopToTopDeck



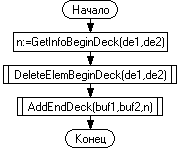
Процедура BottomToBottomDeck



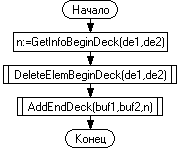
Процедура QuestDeck



Процедура TopToBottomDeck



Процедура BottomToTopDeck



6. Функциональные тесты

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Стек:  1  2  3 | Стек: |
| Стек: | Стек пуст! |
| Стек:  1  50  2 | Стек:  50 |
| Очередь:  1  29  30 | Очередь: |
| Очередь: | Очередь пуста! |
| Очередь:  1  31  31 | Очередь:  31  31 |
| Дек:  3 | Дек:  3 |
| Дек:  3  50  2  3 | Дек:  50  2 |
| Дек:  3  50  1  2  30 | Дек:  50  1  2 |
| Дек: | Дек пуст! |

7. Программа

7.1 Стек (списком)

type

tinfo = integer;

TStack =^PStack;

PStack = record

info:tinfo;

next: tstack;

end;

function IsEmptyStack(const st:tstack):boolean;

procedure AddStack(var st:tstack; const n:tinfo);

procedure TopToTopStack(var buf,st:tstack);

procedure DeleteElemStack(var st:tstack);

procedure DeleteListStack(var st:tstack);

procedure QuestStack(var st:tstack; var fl:boolean);

function IsEndStack(const st:tstack):boolean;

implementation

procedure InitStack(var st:tstack);

begin

st := nil;

end;

function IsEmptyStack(const st:tstack):boolean;

begin

IsEmptyStack := (st=nil);

end;

procedure AddStack(var st:tstack; const n:tinfo);

var k:tstack;

begin

new(k);

k^.info := n;

k^.next := st;

st := k;

end;

procedure DeleteElemStack(var st:tstack);

var k:tstack;

begin

if not(IsEmptyStack(st)) then

begin

k := st;

st := st^.next;

k^.next := nil;

dispose(k);

end;

end;

function GetInfoBeginStack(const st:tstack):tinfo;

begin

GetInfoBeginStack := st^.info;

end;

procedure TopToTopStack(var buf,st:tstack); //

var n:tinfo;

begin

n := GetInfoBeginStack(st);

DeleteElemStack(st);

AddStack(buf,n);

end;

procedure DeleteListStack(var st:tstack); //

begin

while not(IsEmptyStack(st)) do

DeleteElemStack(st);

end;

function IsEndStack(const st:tstack):boolean;

begin

IsEndStack := (st^.next = nil)

end;

procedure QuestStack(var st:tstack; var fl:boolean); //

var buf:tstack;

fl1:boolean;

begin

if IsEmptyStack(st) then

fl := false

else

if (GetInfoBeginStack(st)>=-10) and (GetInfoBeginStack(st)<=30) then

DeleteElemStack(st)

else

begin

InitStack(buf);

fl1 := true;

TopToTopStack(buf,st);

while (fl1) and not(IsEmptyStack(st)) do

begin

if (GetInfoBeginStack(st)>=-10) and (GetInfoBeginStack(st)<=30) then

fl1 := false

else

TopToTopStack(buf,st);

end;

if not(fl1) then

begin

DeleteElemStack(st);

while not(IsEmptyStack(buf)) do

TopToTopStack(st,buf);

end

else

begin

fl := false;

while not(IsEmptyStack(buf)) do

TopToTopStack(st,buf);

end;

end;

end;

//Дополнительные процедуры

procedure AddToEndStack(var st:tstack; n:tinfo); //

var buf:tstack;

begin

InitStack(buf);

while not(IsEmptyStack(st)) do

TopToTopStack(buf,st);

AddStack(st,n);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

procedure DeleteEndStack(var st:tstack); //

var buf:tstack;

begin

if (IsEndStack(st)) then

DeleteElemStack(st)

else

begin

InitStack(buf);

while not(IsEndStack(st)) do

TopToTopStack(buf,st);

DeleteElemStack(st);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

end;

function GetInfoEndStack(var st:tstack): tinfo; //

var buf:tstack;

begin

if (IsEmptyStack(st)) then

GetInfoEndStack := 0

else

begin

InitStack(buf);

while not(IsEndStack(st)) do

TopToTopStack(buf,st);

GetInfoEndStack := GetInfoBeginStack(st);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

end;

procedure AddMiddleStack(var st:tstack;const o,n:tinfo); //

var buf:tstack;

fl:boolean;

begin

InitStack(buf);

fl := true;

while (fl) and not(IsEmptyStack(st)) do

begin

if (GetInfoBeginStack(st) = o) then

fl := false;

TopToTopStack(buf,st);

end;

AddStack(st,n);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

procedure DeleteMiddleStack(var st:tstack; const n:tinfo); //

var buf:tstack;

fl:boolean;

begin

buf := nil;

fl := true;

while (fl) and not(IsEmptyStack(st)) do

begin

if (GetInfoBeginStack(st) = n) then

fl := false

else

TopToTopStack(buf,st);

end;

DeleteElemStack(st);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

end.

Стек (массивом)

type

tinfo = integer;

tstack = record

mas:array of tinfo;

k:integer;

end;

function IsEmptyStack(const st:tstack):boolean;

procedure AddStack(var st:tstack; const n:tinfo);

procedure TopToTopStack(var buf,st:tstack);

procedure DeleteElemStack(var st:tstack);

procedure DeleteListStack(var st:tstack);

procedure QuestStack(var st:tstack; var fl:boolean);

implementation

function IsEmptyStack(const st:tstack):boolean;

begin

IsEmptyStack := (st.k = 0);

end;

function IsEndStack(const st:tstack):boolean;

begin

IsEndStack := (st.k = 1);

end;

procedure AddStack(var st:tstack;const n:tinfo);

var i:integer;

begin

st.k := st.k + 1;

SetLength(st.mas,st.k);

i := st.k-1;

while i<>0 do

begin

st.mas[i] := st.mas[i-1];

i := i-1;

end;

st.mas[0] := n;

end;

procedure DeleteElemStack(var st:tstack);

var i:integer;

begin

i := 0;

if not(IsEmptyStack(st)) then

begin

while (i<>(st.k-1)) do

begin

st.mas[i] := st.mas[i+1];

i := i+1;

end;

st.mas[st.k-1] := 0;

st.k := st.k-1;

SetLength(st.mas,st.k);

end;

end;

procedure DeleteListStack(var st:tstack); //

begin

while not(IsEmptyStack(st)) do

DeleteElemStack(st);

end;

procedure InitStack(var st:tstack);

begin

SetLength(st.mas,0);

st.k := 0;

end;

function GetInfoBeginStack(const st:tstack):tinfo;

begin

GetInfoBeginStack := st.mas[0];

end;

procedure TopToTopStack(var buf,st:tstack); //

var n:tinfo;

begin

n := GetInfoBeginStack(st);

DeleteElemStack(st);

AddStack(buf,n);

end;

procedure QuestStack(var st:tstack; var fl:boolean); //

var buf:tstack;

fl1:boolean;

begin

if IsEmptyStack(st) then

fl := false

else

if (GetInfoBeginStack(st)>=-10) and (GetInfoBeginStack(st)<=30) then

DeleteElemStack(st)

else

begin

InitStack(buf);

fl1 := true;

TopToTopStack(buf,st);

while (fl1) and not(IsEmptyStack(st)) do

begin

if (GetInfoBeginStack(st)>=-10) and (GetInfoBeginStack(st)<=30) then

fl1 := false

else

TopToTopStack(buf,st);

end;

if not(fl1) then

begin

DeleteElemStack(st);

while not(IsEmptyStack(buf)) do

TopToTopStack(st,buf);

end

else

begin

fl := false;

while not(IsEmptyStack(buf)) do

TopToTopStack(st,buf);

end;

end;

end;

//Дополнительные процедуры

procedure AddToEndStack(var st:tstack; n:tinfo); //

var buf:tstack;

begin

InitStack(buf);

while not(IsEmptyStack(st)) do

TopToTopStack(buf,st);

AddStack(st,n);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

procedure DeleteEndStack(var st:tstack); //

var buf:tstack;

begin

if (IsEndStack(st)) then

DeleteElemStack(st)

else

begin

InitStack(buf);

while not(IsEndStack(st)) do

TopToTopStack(buf,st);

DeleteElemStack(st);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

end;

function GetInfoEndStack(var st:tstack): tinfo; //

var buf:tstack;

begin

if (IsEmptyStack(st)) then

GetInfoEndStack := 0

else

begin

InitStack(buf);

while not(IsEndStack(st)) do

TopToTopStack(buf,st);

GetInfoEndStack := GetInfoBeginStack(st);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

end;

procedure AddMiddleStack(var st:tstack;const o,n:tinfo); //

var buf:tstack;

fl:boolean;

begin

InitStack(buf);

fl := true;

while (fl) and not(IsEmptyStack(st)) do

begin

if (GetInfoBeginStack(st) = o) then

fl := false;

TopToTopStack(buf,st);

end;

AddStack(st,n);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

procedure DeleteMiddleStack(var st:tstack; const n:tinfo); //

var buf:tstack;

fl:boolean;

begin

InitStack(buf);

fl := true;

while (fl) and not(IsEmptyStack(st)) do

begin

if (GetInfoBeginStack(st) = n) then

fl := false

else

TopToTopStack(buf,st);

end;

DeleteElemStack(st);

while not(IsEmptyStack(Buf)) do

TopToTopStack(st,buf);

end;

end.

7.2 Очередь (списком)

type

tinfo = integer;

PElem = ^TElem;

TElem = record

info: tinfo;

next: PElem;

end;

TQueue = record

head,tail:PElem;

end;

function IsEmptyQueue(const qu:tqueue):boolean;

procedure AddQueue(var qu:tqueue; const n:tinfo);

procedure DeleteElemQueue( var qu:tqueue);

procedure DeleteListQueue(var qu:tqueue);

procedure TopToBottomQueue(var qu,buf:tqueue);

procedure QuestQueue(var qu:tqueue; var fl:boolean);

implementation

function IsEmptyQueue(const qu:tqueue):boolean;

begin

IsEmptyQueue := (qu.tail=nil);

end;

procedure AddQueue(var qu:tqueue;const n:tinfo);

var k:PElem;

begin

new(k);

k^.info := n;

k^.next := nil;

if IsEmptyQueue(qu) then

begin

qu.head := k;

qu.tail := k;

end

else

begin

qu.tail^.next := k;

qu.tail := k;

end;

end;

procedure DeleteElemQueue(var qu:tqueue);

var k:PElem;

begin

if not(IsEmptyQueue(qu)) then

if qu.head^.next = nil then

begin

qu.head := nil;

qu.tail := nil;

end

else

begin

k := qu.head;

qu.head := qu.head^.next;

k^.next := nil;

dispose(k);

end;

end;

procedure InitQueue(var qu:tqueue);

begin

qu.tail := nil;

end;

procedure DeleteListQueue(var qu:tqueue);

begin

while not(IsEmptyQueue(qu)) do

DeleteElemQueue(qu);

end;

function TakeInfoBeginQueue(const qu:tqueue):tinfo;

begin

TakeInfoBeginQueue := qu.head^.info;

end;

procedure TopToBottomQueue(var qu,buf:tqueue);

var n:tinfo;

begin

n := TakeInfoBeginQueue(qu);

DeleteElemQueue(qu);

AddQueue(buf,n);

end;

procedure QuestQueue(var qu:tqueue;var fl:boolean);

var buf:tqueue;

fl1:boolean;

begin

if IsEmptyQueue(qu) then

fl := false

else

if (TakeInfoBeginQueue(qu)>=-10) and (TakeInfoBeginQueue(qu)<=30) then

DeleteElemQueue(qu)

else

begin

fl1 := true;

InitQueue(buf);

TopToBottomQueue(qu,buf);

while (fl1) and not(IsEmptyQueue(qu)) do

if (TakeInfoBeginQueue(qu)>=-10) and (TakeInfoBeginQueue(qu)<=30) then

fl1 := false

else

TopToBottomQueue(qu,buf);

if not(fl1) then

begin

DeleteElemQueue(qu);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end

else

begin

fl := false;

qu := buf;

end;

end;

end;

//Дополнительные процелуры

procedure AddtoBeginQueue(var qu:tqueue;const n:tinfo);

var buf:tqueue;

begin

AddQueue(buf,n);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end;

procedure AddMiddleQueue(var qu:tqueue;const n:tinfo;const o:tinfo);

var buf:tqueue;

fl : boolean;

begin

InitQueue(buf);

fl := true;

while fl and not(IsEmptyQueue(qu)) do

begin

if TakeInfoBeginQueue(qu) = o then

fl := false;

TopToBottomQueue(qu,buf);

end;

if not(fl) then

begin

AddQueue(buf,n);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end

else

qu := buf;

end;

procedure DeleteEndQueue(var qu:tqueue);

var buf:tqueue;

begin

InitQueue(buf);

while (qu.head<>qu.tail) do

TopToBottomQueue(qu,buf);

DeleteElemQueue(qu);

qu := buf;

end;

procedure DeleteMiddleQueue(var qu:tqueue; const n:tinfo);

var buf:tqueue;

fl : boolean;

begin

if (TakeInfoBeginQueue(qu) = n) then

DeleteElemQueue(qu)

else

begin

InitQueue(buf);

fl := true;

TopToBottomQueue(qu,buf);

while (fl) and not(IsEmptyQueue(qu)) do

if TakeInfoBeginQueue(qu) = n then

fl := false

else

TopToBottomQueue(qu,buf);

if not (fl) then

begin

DeleteElemQueue(qu);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end

else

qu := buf;

end;

end;

function TakeInfoEndQueue(const qu:tqueue):tinfo;

begin

TakeInfoEndQueue := qu.tail^.info;

end;

end.

Очередь (массивом)

type

tinfo = integer;

tqueue = record

mas:array of tinfo;

k:integer;

end;

function IsEmptyQueue(const qu:tqueue):boolean;

procedure AddQueue(var qu:tqueue; const n:tinfo);

procedure DeleteElemQueue(var qu:tqueue);

procedure DeleteListQueue(var qu:tqueue);

procedure TopToBottomQueue(var qu,buf:tqueue);

procedure QuestQueue(var qu:tqueue;var fl:boolean);

implementation

function IsEmptyQueue(const qu:tqueue):boolean;

begin

IsEmptyQueue := (qu.k=0);

end;

function IsOneElemQueue(const qu:tqueue):boolean;

begin

IsOneElemQueue := (qu.k = 1);

end;

procedure AddQueue(var qu:tqueue; const n:tinfo);

begin

qu.k := qu.k+1;

SetLength(qu.mas,qu.k);

qu.mas[qu.k-1] := n;

end;

procedure DeleteElemQueue(var qu:tqueue);

var i:integer;

begin

i := 0;

if not(IsEmptyQueue(qu)) then

begin

while (i<>(qu.k-1)) do

begin

qu.mas[i] := qu.mas[i+1];

i := i+1;

end;

qu.mas[qu.k-1] := 0;

qu.k := qu.k-1;

SetLength(qu.mas,qu.k);

end;

end;

procedure DeleteListQueue(var qu:tqueue);

begin

while not(IsEmptyQueue(qu)) do

DeleteElemQueue(qu);

end;

procedure InitQueue(var qu:tqueue);

begin

SetLength(qu.mas,0);

qu.k := 0;

end;

function TakeInfoBeginQueue(const qu:tqueue):tinfo;

begin

TakeInfoBeginQueue := qu.mas[0];

end;

procedure TopToBottomQueue(var qu,buf:tqueue);

var n:tinfo;

begin

n := TakeInfoBeginQueue(qu);

DeleteElemQueue(qu);

AddQueue(buf,n);

end;

procedure QuestQueue(var qu:tqueue;var fl:boolean);

var buf:tqueue;

fl1:boolean;

begin

if IsEmptyQueue(qu) then

fl := false

else

if (TakeInfoBeginQueue(qu)>=-10) and (TakeInfoBeginQueue(qu)<=30) then

DeleteElemQueue(qu)

else

begin

fl1 := true;

InitQueue(buf);

TopToBottomQueue(qu,buf);

while (fl1) and not(IsEmptyQueue(qu)) do

if (TakeInfoBeginQueue(qu)>=-10) and (TakeInfoBeginQueue(qu)<=30) then

fl1 := false

else

TopToBottomQueue(qu,buf);

if not(fl1) then

begin

DeleteElemQueue(qu);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end

else

begin

fl := false;

qu := buf;

end;

end;

end;

//Дополнительные процедуры

procedure AddtoBeginQueue(var qu:tqueue;const n:tinfo);

var buf:tqueue;

begin

InitQueue(buf);

AddQueue(buf,n);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end;

procedure AddMiddleQueue( var qu:tqueue; const n:tinfo; const o:tinfo);

var buf:tqueue;

fl:boolean;

begin

InitQueue(buf);

fl := true;

while fl and not(IsEmptyQueue(qu)) do

begin

if TakeInfoBeginQueue(qu) = o then

fl := false;

TopToBottomQueue(qu,buf);

end;

if not(fl) then

begin

AddQueue(buf,n);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end

else

qu := buf;

end;

procedure DeleteEndQueue(var qu:tqueue);

var buf:tqueue;

begin

InitQueue(buf);

while not(IsOneElemQueue(qu)) do

TopToBottomQueue(qu,buf);

DeleteElemQueue(qu);

qu := buf;

end;

procedure DeleteMiddleQueue(var qu:tqueue;const n:tinfo);

var buf:tqueue;

fl:boolean;

begin

if TakeInfoBeginQueue(qu) = n then

DeleteElemQueue(qu)

else

begin

InitQueue(buf);

fl := true;

TopToBottomQueue(qu,buf);

while (fl) and not(IsEmptyQueue(qu)) do

if TakeInfoBeginQueue(qu) = n then

fl := false

else

TopToBottomQueue(qu,buf);

if not (fl) then

begin

DeleteElemQueue(qu);

while not(IsEmptyQueue(qu)) do

TopToBottomQueue(qu,buf);

qu := buf;

end

else

qu := buf;

end;

end;

function TakeInfoEndQueue(const qu:tqueue):tinfo;

begin

TakeInfoEndQueue := qu.mas[qu.k-1];

end;

end.

7.3 Дек (списком)

type

tinfo=integer;

TDeck =^PDeck;

PDeck = record

info:tinfo;

next,prev:tdeck;

end;

function IsEmptyDeck(const de1,de2:tdeck):boolean;

procedure AddBeginDeck(var de1,de2:tdeck; const n:tinfo);

procedure AddEndDeck(var de1,de2:tdeck; const n:tinfo);

procedure DeleteElemBeginDeck(var de1,de2:tdeck);

procedure DeleteElemEndDeck(var de1,de2:tdeck);

procedure DeleteListDeck(var de1,de2:tdeck);

procedure TopToTopDeck(var buf1,buf2,de1,de2:tdeck);

procedure BottomToBottomDeck(var buf1,buf2,de1,de2:tdeck);

procedure QuestDeck(var de1,de2:tdeck; var fl:boolean);

implementation

function IsEmptyDeck(const de1,de2:tdeck):boolean;

begin

IsEmptyDeck := (de1=de2) and (de1 = nil);

end;

procedure AddBeginDeck(var de1,de2:tdeck; const n:tinfo);

var k:tdeck;

begin

new(k);

k^.info := n;

if IsEmptyDeck(de1,de2) then

begin

k^.next := nil;

k^.prev := nil;

de1 := k;

de2 := k;

end

else

begin

k^.next := de1;

de1^.prev := k;

k^.prev := nil;

de1 := k;

end;

end;

procedure AddEndDeck(var de1,de2:tdeck; const n:tinfo);

var k:tdeck;

begin

new(k);

k^.info := n;

if IsEmptyDeck(de1,de2) then

begin

k^.next := nil;

k^.prev := nil;

de1 := k;

de2 := k;

end

else

begin

k^.prev := de2;

de2^.next := k;

k^.next := nil;

de2 := k;

end;

end;

procedure DeleteElemBeginDeck(var de1,de2:tdeck);

var k:tdeck;

begin

if not(IsEmptyDeck(de1,de2)) then

begin

if de1=de2 then

begin

de1 := nil;

de2 := nil;

end

else

begin

k := de1;

de1 := de1^.next;

k^.next := nil;

de1^.prev := nil;

dispose(k);

end;

end;

end;

procedure DeleteElemEndDeck(var de1,de2:tdeck);

var k:tdeck;

begin

if not( IsEmptyDeck(de1,de2)) then

begin

if de1=de2 then

begin

de1 := nil;

de2 := nil;

end

else

begin

k := de2;

de2 := de2^.prev;

k^.prev := nil;

de2^.next := nil;

dispose(k);

end;

end;

end;

procedure DeleteListDeck(var de1,de2:tdeck);

begin

while not(IsEmptyDeck(de1,de2)) do

DeleteElemBeginDeck(de1,de2);

end;

function GetInfoBeginDeck(const de1,de2:tdeck):tinfo;

begin

GetInfoBeginDeck := de1^.info;

end;

function GetInfoEndDeck(const de1,de2:tdeck):tinfo;

begin

GetInfoEndDeck := de2^.info;

end;

procedure TopToTopDeck(var buf1,buf2,de1,de2:tdeck);

var n:tinfo;

begin

n := GetInfoBeginDeck(de1,de2);

DeleteElemBeginDeck(de1,de2);

AddBeginDeck(buf1,buf2,n);

end;

procedure BottomToBottomDeck(var buf1,buf2,de1,de2:tdeck);

var n:tinfo;

begin

n := GetInfoEndDeck(de1,de2);

DeleteElemEndDeck(de1,de2);

AddEndDeck(buf1,buf2,n);

end;

procedure InitDeck(var de1,de2:tdeck);

begin

de1 := nil;

de2 := nil;

end;

function IsOneElemDeck(const de1,de2:tdeck):boolean;

begin

IsOneElemDeck := de1=de2;

end;

procedure QuestDeck(var de1,de2:tdeck; var fl:boolean);

var buf1,buf2:tdeck;

fl1:boolean;

begin

if de1 = de2 then

fl := false

else

if ((GetInfoBeginDeck(de1,de2)>=-10) and (GetInfoBeginDeck(de1,de2)<=30)) and ((GetInfoEndDeck(de1,de2)>=-10) and (GetInfoEndDeck(de1,de2)<=30)) then

begin

DeleteElemBeginDeck(de1,de2);

DeleteElemEndDeck(de1,de2);

end

else

begin

fl1 := true;

InitDeck(buf1,buf2);

while (fl1) and not(IsOneElemDeck(de1,de2)) and not(IsEmptyDeck(de1,de2)) do

begin

if ((GetInfoBeginDeck(de1,de2)>=-10) and (GetInfoBeginDeck(de1,de2)<=30)) and ((GetInfoEndDeck(de1,de2)>=-10) and (GetInfoEndDeck(de1,de2)<=30)) then

fl1 := false

else

begin

TopToTopDeck(buf1,buf2,de1,de2);

BottomToBottomDeck(buf1,buf2,de1,de2);

end;

end;

if not(fl1) then

begin

DeleteElemBeginDeck(de1,de2);

DeleteElemEndDeck(de1,de2);

while not(IsEmptyDeck(buf1,buf1)) do

begin

TopToTopDeck(de1,de2,buf1,buf2);

BottomToBottomDeck(de1,de2,buf1,buf2);

end;

end

else

begin

fl := false;

while not(IsEmptyDeck(buf1,buf2)) do

begin

TopToTopDeck(de1,de2,buf1,buf2);

BottomToBottomDeck(de1,de2,buf1,buf2);

end;

end;

end;

end;

//Дополнительные процедуры

procedure BottomToTopDeck(var buf1,buf2,de1,de2:tdeck);

var n:tinfo;

begin

n := GetInfoEndDeck(de1,de2);

DeleteElemEndDeck(de1,de2);

AddBeginDeck(buf1,buf2,n);

end;

procedure TopToBottomDeck(var buf1,buf2,de1,de2:tdeck);

var n:tinfo;

begin

n := GetInfoBeginDeck(de1,de2);

DeleteElemBeginDeck(de1,de2);

AddEndDeck(buf1,buf2,n);

end;

end.

Дек (массивом)

type

tinfo = integer;

tdeck = record

mas:array of tinfo;

k:integer;

end;

function IsEmptyDeck(const de:tdeck):boolean;

procedure AddBeginDeck(var de:tdeck; const n:tinfo);

procedure AddEndDeck(var de:tdeck; const n:tinfo);

procedure DeleteElemBeginDeck(var de:tdeck);

procedure DeleteElemEndDeck(var de:tdeck);

procedure DeleteListDeck(var de:tdeck);

procedure TopToTopDeck(var buf,de:tdeck);

procedure BottomToBottomDeck(var buf,de:tdeck);

procedure QuestDeck(var de:tdeck; var fl:boolean);

implementation

function IsEmptyDeck(const de:tdeck):boolean;

begin

IsEmptyDeck := (de.k=0);

end;

function IsOneElemDeck(const de:tdeck):boolean;

begin

IsOneElemDeck := (de.k = 1);

end;

procedure AddBeginDeck(var de:tdeck;const n:tinfo);

var i:integer;

begin

de.k := de.k + 1;

SetLength(de.mas,de.k);

i := de.k-1;

while i<>0 do

begin

de.mas[i] := de.mas[i-1];

i := i-1;

end;

de.mas[0] := n;

end;

procedure AddEndDeck(var de:tdeck; const n:tinfo);

begin

de.k := de.k+1;

SetLength(de.mas,de.k);

de.mas[de.k-1] := n;

end;

procedure DeleteElemBeginDeck(var de:tdeck);

var i:integer;

begin

i := 0;

if not(IsEmptyDeck(de)) then

begin

while (i<>(de.k-1)) do

begin

de.mas[i] := de.mas[i+1];

i := i+1;

end;

de.mas[de.k-1] := 0;

de.k := de.k-1;

SetLength(de.mas,de.k);

end;

end;

procedure DeleteElemEndDEck(var de:tdeck);

begin

de.mas[de.k-1] := 0;

de.k := de.k - 1;

SetLength(de.mas,de.k);

end;

procedure DeleteListDeck(var de:tdeck);

begin

while not(IsEmptyDeck(de)) do

DeleteElemBeginDeck(de);

end;

procedure InitDeck(var de:tdeck);

begin

SetLength(de.mas,0);

de.k := 0;

end;

function GetInfoBeginDeck(const de:tdeck):tinfo;

begin

GetInfoBeginDeck := de.mas[0];

end;

function GetInfoEndDeck(const de:tdeck):tinfo;

begin

GetInfoEndDeck := de.mas[de.k-1];

end;

procedure TopToTopDeck(var buf,de:tdeck);

var n:tinfo;

begin

n := GetInfoBeginDeck(de);

DeleteElemBeginDeck(de);

AddBeginDeck(buf,n);

end;

procedure BottomToBottomDeck(var buf,de:tdeck);

var n:tinfo;

begin

n := GetInfoEndDeck(de);

DeleteElemEndDeck(de);

AddEndDeck(buf,n);

end;

procedure QuestDeck(var de:tdeck;var fl:boolean);

var buf:tdeck;

fl1:boolean;

begin

if (IsEmptyDeck(de)) then

fl := false

else

if ((GetInfoBeginDeck(de)>=-10) and (GetInfoBeginDeck(de)<=30)) and ((GetInfoEndDeck(de)>=-10) and (GetInfoEndDeck(de)<=30)) then

begin

DeleteElemBeginDeck(de);

DeleteElemEndDeck(de);

end

else

begin

fl1 := true;

InitDeck(buf);

while (fl1) and not(IsOneElemDeck(de)) and not(IsEmptyDeck(de)) do

begin

if ((GetInfoBeginDeck(de)>=-10) and (GetInfoBeginDeck(de)<=30)) and ((GetInfoEndDeck(de)>=-10) and (GetInfoEndDeck(de)<=30)) then

fl1 := false

else

begin

TopToTopDeck(buf,de);

BottomToBottomDeck(buf,de);

end;

end;

if not(fl1) then

begin

DeleteElemBeginDeck(de);

DeleteElemEndDeck(de);

while not(IsEmptyDeck(buf)) do

begin

TopToTopDeck(de,buf);

BottomToBottomDeck(de,buf);

end;

end

else

begin

fl := false;

while not(IsEmptyDeck(buf)) do

begin

TopToTopDeck(de,buf);

BottomToBottomDeck(de,buf);

end;

end;

end;

end;

//Дополнительные процедуры

procedure BottomToTopDeck(var buf,de:tdeck);

var n:tinfo;

begin

n := GetInfoEndDeck(de);

DeleteElemEndDeck(de);

AddBeginDeck(buf,n);

end;

procedure TopToBottomDeck(var buf,de:tdeck);

var n:tinfo;

begin

n := GetInfoBeginDeck(de);

DeleteElemBeginDeck(de);

AddEndDeck(buf,n);

end;

end.

7.4 Основной unit (списки)

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs,Vcl.StdCtrls, Vcl.Menus, Unit2,Unit3,Unit4;

type

mas = array of integer;

structures = (Stack, Queue, Deck);

TForm1 = class(TForm)

OpenDialog: TOpenDialog;

SaveDialog: TSaveDialog;

MainMenu: TMainMenu;

MCondition: TMenuItem;

MChoose: TMenuItem;

MStack: TMenuItem;

MQueue: TMenuItem;

MSaveNLoad: TMenuItem;

MSave: TMenuItem;

MQuest: TMenuItem;

MDeck: TMenuItem;

MLoad: TMenuItem;

MExit: TMenuItem;

GroupBoxStack: TGroupBox;

ListBoxStack: TListBox;

Label1: TLabel;

EditElemStack: TEdit;

ButtonAddStack: TButton;

GroupBoxQueue: TGroupBox;

Label2: TLabel;

ListBoxQueue: TListBox;

EditElemQueue: TEdit;

ButtonAddQueue: TButton;

GroupBoxDeck: TGroupBox;

ButtonDeleteElemQueue: TButton;

ButtonDeleteListQueue: TButton;

ButtonDeleteElemStack: TButton;

ButtonDeleteListStack: TButton;

ListBoxDeck2: TListBox;

Label3: TLabel;

EditElemDeck: TEdit;

ButtonAddBeginDeck: TButton;

ButtonAddEndDeck: TButton;

ButtonDeleteBeginDeck: TButton;

ButtonDeleteEndDeck: TButton;

ButtonDeleteListDeck: TButton;

ListBoxDeck1: TListBox;

procedure MExitClick(Sender: TObject);

procedure MConditionClick(Sender: TObject);

procedure EditElemStackChange(Sender: TObject);

procedure ButtonAddStackClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure MQuestClick(Sender: TObject);

procedure MStackClick(Sender: TObject);

procedure MQueueClick(Sender: TObject);

procedure ButtonAddQueueClick(Sender: TObject);

procedure ButtonDeleteElemQueueClick(Sender: TObject);

procedure ButtonDeleteListQueueClick(Sender: TObject);

procedure ButtonDeleteElemStackClick(Sender: TObject);

procedure ButtonDeleteListStackClick(Sender: TObject);

procedure EditElemQueueChange(Sender: TObject);

procedure EditElemDeckChange(Sender: TObject);

procedure ButtonAddBeginDeckClick(Sender: TObject);

procedure ButtonAddEndDeckClick(Sender: TObject);

procedure ButtonDeleteBeginDeckClick(Sender: TObject);

procedure ButtonDeleteEndDeckClick(Sender: TObject);

procedure ButtonDeleteListDeckClick(Sender: TObject);

procedure MDeckClick(Sender: TObject);

procedure MSaveClick(Sender: TObject);

procedure MLoadClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

InUse:Structures;

st : tstack;

qu : tqueue;

de1,de2 : tdeck;

implementation

{$R \*.dfm}

procedure inputmas(var x:mas; var f:textfile; var k:integer);

var i:integer;

begin

i := 0;

while not (eof(f)) do

begin

SetLength(x,i+1);

readln(f,x[i]);

i := i+1;

end;

k := i;

end;

procedure ShowListStack(const st:tstack; listbox:tlistbox);

var k:tstack;

begin

ListBox.Clear;

k := st;

while (k<>nil) do

begin

ListBox.Items.Add(inttostr(k^.info));

k := k^.next;

end;

end;

procedure inputstack(const x:mas;const k:integer; var st:tstack);

var inform:tinfo;

i:integer;

begin

i := k-1;

while (i<>-1) do

begin

inform := x[i];

AddStack(st,inform);

i := i-1;

end;

end;

procedure outputstack(var f:textfile; const st:tstack);

var k:tstack;

begin

k := st;

while (k<>nil) do

begin

writeln(f,k^.info);

k := k^.next;

end;

end;

procedure ShowListQueue(const qu:tqueue; var listbox:tlistbox);

var k:tqueue;

begin

ListBox.Clear;

k := qu;

while (k.head<>nil) do

begin

ListBox.Items.Add(inttostr(k.head^.info));

k.head := k.head^.next;

end;

end;

procedure inputqueue(const x:mas;const k:integer; var qu:tqueue);

var inform:tinfo;

i:integer;

begin

i := 0;

while (i<>k) do

begin

inform := x[i];

AddQueue(qu,inform);

i := i+1;

end;

end;

procedure outputqueue(var f:textfile; const qu:tqueue);

var k:tqueue;

begin

k := qu;

while (k.head<>nil) do

begin

writeln(f,k.head^.info);

k.head := k.head^.next;

end;

end;

procedure ShowListDeck1(const de1,de2:tdeck; var listbox:tlistbox);

var k:tdeck;

begin

ListBox.Clear;

k := de1;

while (k<>nil) do

begin

ListBox.Items.Add(inttostr(k^.info));

k := k^.next;

end;

end;

procedure ShowListDeck2(const de1,de2:tdeck; var listbox:tlistbox);

var k:tdeck;

begin

ListBox.Clear;

k := de2;

while (k<>nil) do

begin

ListBox.Items.Add(inttostr(k^.info));

k := k^.prev;

end;

end;

procedure outputdeck(var f:textfile; const de1,de2:tdeck);

var a:tdeck;

begin

a := de1;

while (a<>nil) do

begin

writeln(f, a^.info);

a := a^.next;

end;

end;

procedure inputdeck(const x:mas; const k:integer; var de1,de2:tdeck);

var i:integer;

inform:tinfo;

begin

i := 0;

while (i<>k) do

begin

inform := x[i];

AddEndDeck(de1,de2,inform);

i := i+1;

end;

end;

// Кнопки

procedure TForm1.ButtonAddBeginDeckClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemDeck.Text);

AddBeginDeck(de1,de2,n);

ShowListDeck1(de1,de2,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonAddEndDeckClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemDeck.Text);

AddEndDeck(de1,de2,n);

ShowListDeck1(de1,de2,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonAddQueueClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemQueue.Text);

AddQueue(qu,n);

ShowListQueue(qu,listboxqueue);

EditElemQueue.SetFocus;

end;

procedure TForm1.ButtonAddStackClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemStack.Text);

AddStack(st,n);

ShowListStack(st,listboxstack);

EditElemStack.SetFocus;

end;

procedure TForm1.ButtonDeleteBeginDeckClick(Sender: TObject);

begin

if IsEmptyDeck(de1,de2) then

ShowMessage('Дек пустой')

else

DeleteElemBeginDeck(de1,de2);

ShowListDeck1(de1,de2,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonDeleteElemQueueClick(Sender: TObject);

begin

if IsEmptyQueue(qu) then

ShowMessage('Очередь пустая')

else

DeleteElemQueue(qu);

ShowListQueue(qu,listboxqueue);

EditElemQueue.SetFocus;

end;

procedure TForm1.ButtonDeleteElemStackClick(Sender: TObject);

begin

if IsEmptyStack(st) then

ShowMessage('Стек пуст')

else

DeleteElemStack(st);

ShowListStack(st,listboxstack);

EditElemStack.SetFocus;

end;

procedure TForm1.ButtonDeleteEndDeckClick(Sender: TObject);

begin

if IsEmptyDeck(de1,de2) then

ShowMessage('Дек пустой')

else

DeleteElemEndDeck(de1,de2);

ShowListDeck1(de1,de2,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonDeleteListDeckClick(Sender: TObject);

begin

if IsEmptyDeck(de1,de2) then

ShowMessage('Дек пустой')

else

DeleteListDeck(de1,de2);

ShowListDeck1(de1,de2,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonDeleteListQueueClick(Sender: TObject);

begin

if IsEmptyQueue(qu) then

ShowMessage('Очередь пустая')

else

DeleteListQueue(qu);

ShowListQueue(qu,listboxqueue);

EditElemQueue.SetFocus;

end;

procedure TForm1.ButtonDeleteListStackClick(Sender: TObject);

begin

if IsEmptyStack(st) then

ShowMessage('Стек пуст')

else

DeleteListStack(st);

ShowListStack(st,listboxstack);

EditElemStack.SetFocus;

end;

procedure TForm1.EditElemDeckChange(Sender: TObject);

begin

if (strtoint(EditElemDeck.Text)<-20) and (strtoint(EditElemDeck.Text)>50) then

EditElemDeck.Text := ''

else

begin

ButtonAddBeginDeck.Enabled := true;

ButtonAddEndDeck.Enabled := true;

end;

end;

procedure TForm1.EditElemQueueChange(Sender: TObject);

begin

if (strtoint(EditElemQueue.Text)<-20) and (strtoint(EditElemQueue.Text)>50) then

EditElemQueue.Text := ''

else

begin

ButtonAddQueue.Enabled := true;

end;

end;

procedure TForm1.EditElemStackChange(Sender: TObject);

begin

if (strtoint(EditElemStack.Text)<-20) and (strtoint(EditElemStack.Text)>50) then

EditElemStack.Text := ''

else

begin

ButtonAddStack.Enabled := true;

end;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

InUse := Stack;

GroupBoxStack.Visible := True;

GroupBoxQueue.Visible := False;

GroupBoxDeck.Visible := False;

MStack.Checked := true;

end;

procedure TForm1.MConditionClick(Sender: TObject);

begin

Application.MessageBox('Из стека/очереди удалить эл-ты, попадающие в диапазон [-10..30]'#13#10'Из дека удалить эл-ты, попадающие в диапазон [-10..30], при условии, что они находятся на одинаковом расстоянии от краев дека',

'Условие', MB\_OK or MB\_ICONINFORMATION);

end;

procedure TForm1.MDeckClick(Sender: TObject);

begin

if InUse = Deck then

exit;

MDeck.Checked := true;

if InUse = Stack then

begin

ListBoxStack.Clear;

DeleteListStack(st);

MStack.Checked := false;

GroupBoxStack.Visible := false;

GroupBoxDeck.Visible := true;

InUse := Deck;

end

else

begin

ListBoxQueue.Clear;

DeleteListQueue(qu);

MQueue.Checked := false;

GroupBoxQueue.Visible := false;

GroupBoxDeck.Visible := true;

InUse := Deck;

end;

end;

procedure TForm1.MExitClick(Sender: TObject);

begin

Close

end;

procedure TForm1.MLoadClick(Sender: TObject);

var x:mas;

f:textfile;

k:integer;

begin

OpenDialog.InitialDir := GetCurrentDir;

if OpenDialog.Execute then

begin

try

AssignFile(f, OpenDialog.FileName);

Reset(f);

except

Application.MessageBox(PWideChar('Невозможно открыть файл для чтения'),

'Ошибка', MB\_OK or MB\_ICONWARNING);

Exit;

end;

inputmas(x,f,k);

if InUse = deck then

begin

inputdeck(x,k,de1,de2);

ShowListDeck1(de1,de1,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

end

else

if InUse=Stack then

begin

inputstack(x,k,st);

ShowListStack(st,listboxstack);

end

else

begin

inputqueue(x,k,qu);

ShowlistQueue(qu,listboxqueue);

end;

CloseFile(f);

end;

end;

procedure TForm1.MQuestClick(Sender: TObject);

var fl:boolean;

begin

fl := true;

if InUse = Stack then

begin

if IsEmptyStack(st) then

ShowMessage('Стек пуст')

else

begin

while fl do

QuestStack(st,fl);

ShowListStack(st,listboxstack);

end;

end

else

if InUse = Queue then

begin

if IsEmptyQueue(qu) then

ShowMessage('Очередь пуста')

else

begin

while fl do

QuestQueue(qu,fl);

ShowListQueue(qu,listboxqueue);

end;

end

else

begin

if IsEmptyDeck(de1,de2) then

ShowMessage('Дек пуст')

else

begin

while fl do

QuestDeck(de1,de2,fl);

ShowListDeck1(de1,de2,listboxdeck1);

ShowListDeck2(de1,de2,listboxdeck2);

end;

end;

end;

procedure TForm1.MQueueClick(Sender: TObject);

begin

if Inuse = Queue then

exit;

MQueue.Checked := true;

if InUse = Stack then

begin

ListBoxStack.Clear;

DeleteListStack(st);

MStack.Checked := false;

GroupBoxStack.Visible := false;

GroupBoxQueue.Visible := true;

InUse := Queue;

end

else

begin

ListBoxDeck1.Clear;

ListBoxDeck2.Clear;

DeleteListDeck(de1,de2);

MDeck.Checked := false;

GroupBoxDeck.Visible := false;

GroupBoxQueue.Visible := true;

InUse := Queue;

end;

end;

procedure TForm1.MSaveClick(Sender: TObject);

var f:TextFile;

begin

SaveDialog.InitialDir := GetCurrentDir;

if SaveDialog.Execute then

begin

try

AssignFile(f, SaveDialog.FileName);

rewrite(f);

except

Application.MessageBox('Невозможно открыть указанный файл для записи', 'Ошибка', MB\_OK or MB\_ICONWARNING);

Exit;

end;

if InUse = Stack then

outputstack(f,st)

else

if InUse = Queue then

outputqueue(f,qu)

else

outputdeck(f,de1,de2);

Closefile(f);

end;

end;

procedure TForm1.MStackClick(Sender: TObject);

begin

if InUse = Stack then

exit;

MStack.Checked := true;

if InUse = Queue then

begin

ListBoxQueue.Clear;

DeleteListQueue(qu);

MQueue.Checked := false;

GroupBoxQueue.Visible := false;

GroupBoxStack.Visible := true;

InUse := Stack;

end

else

begin

ListBoxDeck1.Clear;

ListBoxDeck2.Clear;

DeleteListDeck(de1,de2);

MDeck.Checked := false;

GroupBoxDeck.Visible := false;

GroupBoxStack.Visible := true;

InUse := Stack;

end;

end;

end.

Основной unit (массив)

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs,Vcl.StdCtrls, Vcl.Menus,unit2,unit3,unit4;

type

structures = (Stack,Queue,Deck);

TForm1 = class(TForm)

GroupBoxStack: TGroupBox;

Label1: TLabel;

ListBoxStack: TListBox;

EditElemStack: TEdit;

ButtonAddStack: TButton;

ButtonDeleteElemStack: TButton;

ButtonDeleteListStack: TButton;

OpenDialog: TOpenDialog;

SaveDialog: TSaveDialog;

MainMenu: TMainMenu;

MCondition: TMenuItem;

MChoose: TMenuItem;

MStack: TMenuItem;

MQueue: TMenuItem;

MDeck: TMenuItem;

MSaveNLoad: TMenuItem;

MSave: TMenuItem;

MLoad: TMenuItem;

MQuest: TMenuItem;

MExit: TMenuItem;

GroupBoxDeck: TGroupBox;

Label3: TLabel;

ListBoxDeck2: TListBox;

EditElemDeck: TEdit;

ButtonAddBeginDeck: TButton;

ButtonAddEndDeck: TButton;

ButtonDeleteBeginDeck: TButton;

ButtonDeleteEndDeck: TButton;

ButtonDeleteListDeck: TButton;

ListBoxDeck1: TListBox;

GroupBoxQueue: TGroupBox;

Label2: TLabel;

ListBoxQueue: TListBox;

EditElemQueue: TEdit;

ButtonAddQueue: TButton;

ButtonDeleteElemQueue: TButton;

ButtonDeleteListQueue: TButton;

procedure ButtonAddBeginDeckClick(Sender: TObject);

procedure ButtonAddEndDeckClick(Sender: TObject);

procedure ButtonAddQueueClick(Sender: TObject);

procedure ButtonAddStackClick(Sender: TObject);

procedure ButtonDeleteBeginDeckClick(Sender: TObject);

procedure ButtonDeleteElemQueueClick(Sender: TObject);

procedure ButtonDeleteElemStackClick(Sender: TObject);

procedure ButtonDeleteEndDeckClick(Sender: TObject);

procedure ButtonDeleteListDeckClick(Sender: TObject);

procedure ButtonDeleteListQueueClick(Sender: TObject);

procedure ButtonDeleteListStackClick(Sender: TObject);

procedure EditElemDeckChange(Sender: TObject);

procedure EditElemQueueChange(Sender: TObject);

procedure EditElemStackChange(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure MConditionClick(Sender: TObject);

procedure MDeckClick(Sender: TObject);

procedure MExitClick(Sender: TObject);

procedure MLoadClick(Sender: TObject);

procedure MQuestClick(Sender: TObject);

procedure MQueueClick(Sender: TObject);

procedure MSaveClick(Sender: TObject);

procedure MStackClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

InUse: structures;

st : tstack;

qu : tqueue;

de : tdeck;

implementation

{$R \*.dfm}

procedure ShowListStack(const st:tstack; var listbox:tlistbox);

var i:integer;

begin

ListBox.Clear;

i := 0;

while i<>st.k do

begin

ListBox.Items.Add(inttostr(st.mas[i]));

i := i+1;

end;

end;

procedure inputstack(var st:tstack; var f:textfile);

var i:integer;

begin

i := 0;

while not(eof(f)) do

begin

st.k := i+1;

SetLength(st.mas,st.k);

readln(f,st.mas[i]);

i := i+1;

end;

end;

procedure outputstack(const st:tstack; var f:textfile);

var i:integer;

begin

for i := 0 to (st.k-1) do

writeln(f,st.mas[i]);

end;

procedure ShowListQueue(const qu:tqueue; var listbox:tlistbox);

var i:integer;

begin

ListBox.Clear;

i := 0;

while i<>qu.k do

begin

ListBox.Items.Add(inttostr(qu.mas[i]));

i := i+1;

end;

end;

procedure inputqueue(var qu:tqueue; var f:textfile);

var i:integer;

begin

i := 0;

while not(eof(f)) do

begin

qu.k := i+1;

SetLength(qu.mas,qu.k);

readln(f,qu.mas[i]);

i := i+1;

end;

end;

procedure outputqueue(const qu:tqueue; var f:textfile);

var i:integer;

begin

for i := 0 to (qu.k-1) do

writeln(f,qu.mas[i]);

end;

procedure ShowListDeck1(const de:tdeck; var listbox:tlistbox);

var i:integer;

begin

ListBox.Clear;

i := 0;

while i<>de.k do

begin

ListBox.Items.Add(inttostr(de.mas[i]));

i := i+1;

end;

end;

procedure ShowListDeck2(const de:tdeck; var listbox:tlistbox);

var i:integer;

begin

ListBox.Clear;

i := de.k-1;

while i<>-1 do

begin

ListBox.Items.Add(inttostr(de.mas[i]));

i := i-1;

end;

end;

procedure inputdeck(var de:tdeck; var f:textfile);

var i:integer;

begin

i := 0;

while not(eof(f)) do

begin

de.k := i+1;

SetLength(de.mas,de.k);

readln(f,de.mas[i]);

i := i+1;

end;

end;

procedure outputdeck(const de:tdeck; var f:textfile);

var i:integer;

begin

for i := 0 to (de.k-1) do

writeln(f,de.mas[i]);

end;

//Кнопки

procedure TForm1.ButtonAddBeginDeckClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemDeck.Text);

AddBeginDeck(de,n);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonAddEndDeckClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemDeck.Text);

AddEndDeck(de,n);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

EditElemDeck.SetFocus

end;

procedure TForm1.ButtonAddQueueClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemQueue.Text);

AddQueue(qu,n);

ShowListQueue(qu,listboxqueue);

EditElemQueue.SetFocus;

end;

procedure TForm1.ButtonAddStackClick(Sender: TObject);

var n:tinfo;

begin

n := strtoint(EditElemStack.Text);

AddStack(st,n);

ShowListStack(st,listboxstack);

EditElemStack.SetFocus;

end;

procedure TForm1.ButtonDeleteBeginDeckClick(Sender: TObject);

begin

if IsEmptyDeck(de) then

ShowMessage('Дек пустой')

else

DeleteElemBeginDeck(de);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonDeleteElemQueueClick(Sender: TObject);

begin

if IsEmptyQueue(qu) then

ShowMessage('Очередь пустая')

else

DeleteElemQueue(qu);

ShowListQueue(qu,listboxqueue);

EditElemQueue.SetFocus;

end;

procedure TForm1.ButtonDeleteElemStackClick(Sender: TObject);

begin

if IsEmptyStack(st) then

ShowMessage('Стек пуст')

else

DeleteElemStack(st);

ShowListStack(st,listboxstack);

EditElemStack.SetFocus;

end;

procedure TForm1.ButtonDeleteEndDeckClick(Sender: TObject);

begin

if IsEmptyDeck(de) then

ShowMessage('Дек пустой')

else

DeleteElemEndDeck(de);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonDeleteListDeckClick(Sender: TObject);

begin

if IsEmptyDeck(de) then

ShowMessage('Дек пустой')

else

DeleteListDeck(de);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

EditElemDeck.SetFocus;

end;

procedure TForm1.ButtonDeleteListQueueClick(Sender: TObject);

begin

if IsEmptyQueue(qu) then

ShowMessage('Очередь пустая')

else

DeleteListQueue(qu);

ShowListQueue(qu,listboxqueue);

EditElemQueue.SetFocus;

end;

procedure TForm1.ButtonDeleteListStackClick(Sender: TObject);

begin

if IsEmptyStack(st) then

ShowMessage('Стек пуст')

else

DeleteListStack(st);

ShowListStack(st,listboxstack);

EditElemStack.SetFocus;

end;

procedure TForm1.EditElemDeckChange(Sender: TObject);

begin

if (strtoint(EditElemDeck.Text)<-20) and (strtoint(EditElemDeck.Text)>50) then

EditElemDeck.Text := ''

else

begin

ButtonAddBeginDeck.Enabled := true;

ButtonAddEndDeck.Enabled := true;

end;

end;

procedure TForm1.EditElemQueueChange(Sender: TObject);

begin

if (strtoint(EditElemQueue.Text)<-20) and (strtoint(EditElemQueue.Text)>50) then

EditElemQueue.Text := ''

else

begin

ButtonAddQueue.Enabled := true;

end;

end;

procedure TForm1.EditElemStackChange(Sender: TObject);

begin

if (strtoint(EditElemStack.Text)<-20) and (strtoint(EditElemStack.Text)>50) then

EditElemStack.Text := ''

else

begin

ButtonAddStack.Enabled := true;

end;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

InUse := Stack;

GroupBoxStack.Visible := True;

GroupBoxQueue.Visible := False;

GroupBoxDeck.Visible := False;

MStack.Checked := true;

end;

procedure TForm1.MConditionClick(Sender: TObject);

begin

Application.MessageBox('Из стека/очереди удалить эл-ты, попадающие в диапазон [-10..30]'#13#10'Из дека удалить эл-ты, попадающие в диапазон [-10..30], при условии, что они находятся на одинаковом расстоянии от краев дека',

'Условие', MB\_OK or MB\_ICONINFORMATION);

end;

procedure TForm1.MDeckClick(Sender: TObject);

begin

if InUse = Deck then

exit;

MDeck.Checked := true;

if InUse = Stack then

begin

ListBoxStack.Clear;

DeleteListStack(st);

MStack.Checked := false;

GroupBoxStack.Visible := false;

GroupBoxDeck.Visible := true;

InUse := Deck;

end

else

begin

ListBoxQueue.Clear;

DeleteListQueue(qu);

MQueue.Checked := false;

GroupBoxQueue.Visible := false;

GroupBoxDeck.Visible := true;

InUse := Deck;

end;

end;

procedure TForm1.MExitClick(Sender: TObject);

begin

Close

end;

procedure TForm1.MLoadClick(Sender: TObject);

var f:textfile;

begin

OpenDialog.InitialDir := GetCurrentDir;

if OpenDialog.Execute then

begin

try

AssignFile(f, OpenDialog.FileName);

Reset(f);

except

Application.MessageBox(PWideChar('Невозможно открыть файл для чтения'),

'Ошибка', MB\_OK or MB\_ICONWARNING);

Exit;

end;

if InUse = deck then

begin

inputdeck(de,f);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

end

else

if InUse=Stack then

begin

inputstack(st,f);

ShowListStack(st,listboxstack);

end

else

begin

inputqueue(qu,f);

ShowlistQueue(qu,listboxqueue);

end;

CloseFile(f);

end;

end;

procedure TForm1.MQuestClick(Sender: TObject);

var fl:boolean;

begin

fl := true;

if InUse = Stack then

begin

if IsEmptyStack(st) then

ShowMessage('Стек пуст')

else

begin

while fl do

QuestStack(st,fl);

ShowListStack(st,listboxstack);

end;

end

else

if InUse = Queue then

begin

if IsEmptyQueue(qu) then

ShowMessage('Очередь пуста')

else

begin

while fl do

QuestQueue(qu,fl);

ShowListQueue(qu,listboxqueue);

end;

end

else

begin

if IsEmptyDeck(de) then

ShowMessage('Дек пуст')

else

begin

while fl do

QuestDeck(de,fl);

ShowListDeck1(de,listboxdeck1);

ShowListDeck2(de,listboxdeck2);

end;

end;

end;

procedure TForm1.MQueueClick(Sender: TObject);

begin

if Inuse = Queue then

exit;

MQueue.Checked := true;

if InUse = Stack then

begin

ListBoxStack.Clear;

DeleteListStack(st);

MStack.Checked := false;

GroupBoxStack.Visible := false;

GroupBoxQueue.Visible := true;

InUse := Queue;

end

else

begin

ListBoxDeck1.Clear;

ListBoxDeck2.Clear;

DeleteListDeck(de);

MDeck.Checked := false;

GroupBoxDeck.Visible := false;

GroupBoxQueue.Visible := true;

InUse := Queue;

end;

end;

procedure TForm1.MSaveClick(Sender: TObject);

var f:textfile;

begin

SaveDialog.InitialDir := GetCurrentDir;

if SaveDialog.Execute then

begin

try

AssignFile(f, SaveDialog.FileName);

rewrite(f);

except

Application.MessageBox('Невозможно открыть указанный файл для записи', 'Ошибка', MB\_OK or MB\_ICONWARNING);

Exit;

end;

if InUse = Stack then

outputstack(st,f)

else

if InUse = Queue then

outputqueue(qu,f)

else

outputdeck(de,f);

Closefile(f);

end;

end;

procedure TForm1.MStackClick(Sender: TObject);

begin

if InUse = Stack then

exit;

MStack.Checked := true;

if InUse = Queue then

begin

ListBoxQueue.Clear;

DeleteListQueue(qu);

MQueue.Checked := false;

GroupBoxQueue.Visible := false;

GroupBoxStack.Visible := true;

InUse := Stack;

end

else

begin

ListBoxDeck1.Clear;

ListBoxDeck2.Clear;

DeleteListDeck(de);

MDeck.Checked := false;

GroupBoxDeck.Visible := false;

GroupBoxStack.Visible := true;

InUse := Stack;

end;

end;

end.