Coursera, peer-graded assignment for Week 10

# TaskДля сформированного списка решить следующую задачу: удалить из списка все нечетные числа.

# Source code

|  |
| --- |
| *{*  *program.pas*    *Copyright 2017 Pavel Shlyak <shlyak-pavel2011@ya.ru>*  *Based on original source code by Erokhina E. A. <eerokhina@hse.ru>*    *This program is free software; you can redistribute it and/or modify*  *it under the terms of the GNU General Public License as published by*  *the Free Software Foundation; either version 2 of the License, or*  *(at your option) any later version.*    *This program is distributed in the hope that it will be useful,*  *but WITHOUT ANY WARRANTY; without even the implied warranty of*  *MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the*  *GNU General Public License for more details.*    *You should have received a copy of the GNU General Public License*  *along with this program; if not, write to the Free Software*  *Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,*  *MA 02110-1301, USA.*  *}*  *Program Week12;*  *Type*  *recdyn = ^node;*  *node = Record*  *info: integer;*  *next: recdyn;*  *End;*  *Var first: recdyn;*  *Procedure queue(Var first: RecDyn); {создание линейной очереди}*  *Var N: integer;*  *Last, Curr: RecDyn;*  *Begin*  *writeln('Введите натуральные числа:');*  *read(N); {ввод первого числа}*  *If N <= 0 Then*  *First := Nil {занесение признака пустой очереди}*  *Else*  *Begin*  *New(Curr); {выделение памяти для первой записи}*  *First := Curr; {запоминание адреса первой записи очереди}*  *Repeat {цикл ввода и запоминания положительных чисел}*  *Last := Curr; {запоминание адреса последней записи}*  *New(Curr); { выделение памяти для следующей записи}*  *Last^.next := curr;*  *{занесение в поле указателя последней записи очереди адреса*  *следующей записи, которой может и не быть}*  *Last^.Info := N;*  *{занесение введенного числа в информационное поле*  *последней записи}*  *read(N);*  *Until N <= 0;*  *{формирование очереди заканчивается при вводе отрицательного*  *числа или нуля}*  *Last^.next := Nil;*  *{занесение в поле указателя последней записи признака конца очереди}*  *Dispose(Curr);*  *{освобождение памяти для неиспользованной записи}*  *End;*  *End;*  *Procedure deletenumber(Var first:RecDyn);*  *Var curr, pred : RecDyn;*  *Begin*  *curr := first;*  *pred := first;*  *While (curr <> Nil) Do*  *If (curr^.info mod 2)<>0 Then*  *If (curr = first) Then*  *Begin*  *curr := curr^.next;*  *dispose(first);*  *first := curr;*  *pred := first;*  *End*  *Else*  *Begin*  *pred^.next := curr^.next;*  *dispose(curr);*  *curr := pred^.next;*  *End*  *Else {Элемент не удаляется – движение по списку.}*  *Begin*  *pred := curr;*  *curr := curr^.next;*  *End;*  *End;*  *Procedure print(first:RecDyn); {вывод списка на экран}*  *Var curr: RecDyn;*  *Begin*  *If first=Nil Then*  *writeln('список пуст')*  *Else*  *Begin*  *curr := first;*  *While curr<>Nil Do*  *Begin*  *write(curr^.info:5); {вывод поля данных}*  *curr := curr^.next; {переход к следующей записи списка}*  *End;*  *writeln;*  *End;*  *End;*  *Begin*  *writeln('формирование исходного списка (очередь)');*  *queue(first);*  *writeln('исходный список (очередь)');*  *print(first);*  *writeln('список после удаления');*  *deletenumber(first);*  *print(first);*  *End.* |

# Tests

# 