Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по лабораторной работе №12

Тема работы: Динамические структуры

Выполнил

студент: гр. 851003 Распопов П.В.

Проверил: Фадеева Е.П.

Минск 2019

Содержание

[1 Постановка задачи 3](#_Toc10330252)

[2 Описание алгоритмов решения задачи 4](#_Toc10330253)

[3 Структура данных 5](#_Toc10330254)

[4 Схема алгоритма решения задачи по ГОСТ 19.701-90 8](#_Toc10330255)

[Приложение А 13](#_Toc10330256)

[Приложение Б 19](#_Toc10330257)

# Постановка задачи

Даны натуральное число N, действительные числа А1…А2N. Выяснить, верно ли, что для i=1,…,N выполнено:

Ai=2\*A(N-I+1)+A(2\*N-I+1).

Для того чтобы условие было верно, должно выполняться условие:

A1=…=AN= -A(N+1)=…= -A2N;

# Описание алгоритмов решения задачи

Таблица 1 – Описание алгоритмов решения задачи

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предполагаемый тип реализации |
| Create  (FList, aFileName) | Открывает или создает типизированный файл и инициализирует двунаправленный список (FList) путем считывания из файла | FList – получает адрес;  Res – получает адрес с защитой | Процедура |
| Destroy(FList) | Сохраняет двунаправленный список (FList) в типизированный файл и закрывает файл | FList – получает адрес | Процедура |
| getNodeByIndex (FList, i, Res) | Возвращает узел двунаправленного списка (FList) по его индексу (i) | FList – получает адрес;  i – получает адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция.  Res – возвращаемый функцией параметр (узел двунаправленного списка) |
| removeNode  (FList, node) | Удаляет выбранный узел (node) из двунаправленного списка (FList) | FList – получает адрес;  node – получает адрес с защитой | Процедура |
| addItem(FList, item) | Добавляет новую запись (item) в двунаправленный список (FList) | FList – получает адрес;  item – получает адрес с защитой | Процедура |

# Структура данных

Таблица 2 – Структура типов основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TData | record  Number: Integer;  end | Тип-запись, предназначенный для представления информации о числе в последовательности.  Number – действительное число |
| PDate | ^TData | Тип-указатель, предназначенный для числа в последовательности |
| TListNode | class  item: PDate;  next: TListNode;  prev: TListNode;  end | Тип-класс, предназначенный для реализации двунаправленного списка для последовательности из действительных чисел.  item – указатель на элемент в списке;  next – следующий узел списка  prev – предыдущий узел списка |
| TList | class  head: TListNode;  tail: TListNode;  fileName: string;  fCount: Integer;  end | Тип-класс, предназначенный для хранения первого и последнего узла двунаправленного списка, имени типизированного файла и количества узлов в двунаправленном списке |

Таблица 3 – Структура данных основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| FList | TList | Создает или открывает типизированный файл и инициализирует двунаправленный список путем считывания из файла.  Отображает двунаправленный список последовательности действительных чисел. |

Таблица 4 – Структура данных алгоритма Create (FList, aFileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FList | TList | Двунаправленный список | Формальный (параметр-переменная, по адресу) |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| aFileName | String | Имя типизированного файла | Формальный (параметр-переменная, по адресу с защитой) |
| f | File of TData | Типизированный файл | Локальный |
| itemTmp | PData | Указатель на элемент в двунаправленном списке | Локальный |

Таблица 5 – Структура данных алгоритма Destroy (FList)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FList | TList | Двунаправленный список | Формальный (параметр-переменная, по адресу) |
| f | File of TData | Типизированный файл | Локальный |
| nodeCurr | TListNode | Текущий узел в двунаправленном списке | Локальный |

Таблица 6 – Структура данных алгоритма getNodeByIndex (FList, i, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FList | TList | Двунаправленный список | Формальный (параметр-переменная, по адресу) |
| i | Integer | Индекс узла, которого нужно найти | Формальный (параметр-переменная, по адресу с защитой) |
| Res | TListNode | Полученный в результате поиска узел | Формальный (параметр-переменная, по адресу) |
| j | Integer | Счетчик цикла | Локальный |

Таблица 7 – Структура данных алгоритма removeNode (FList, node)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FList | TList | Двунаправленный список | Формальный (параметр-переменная, по адресу) |
| Node | TListNode | Выбранный узел в двунаправленном списке, который нужно удалить | Формальный (параметр-переменная, по адресу с защитой) |

Таблица 8 – Структура данных алгоритма addItem (FList, item)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FList | TList | Двунаправленный список | Формальный (параметр-переменная, по адресу) |
| item | PData | Указатель на элемент, который нужно добавить в список | Формальный (параметр-переменная, по адресу с защитой) |
| newNode | TListNode | Новый узел списка | Локальный |
| nodeCurr | TListNode | Текущий узел списка | Локальный |

# Схема алгоритма решения задачи по ГОСТ 19.701-90



Рисунок 1 – Схема алгоритма Create



Рисунок 2 – Схема алгоритма Destroy



Рисунок 3 – Схема алгоритма getNodeByIndex



Рисунок 4 – Схема алгоритма removeNode



Рисунок 5 – Схема алгоритма addItem

К стр. 11

Приложение А

(обязательное)

Исходный код программы

program lab12;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

SysUtils,

UnitList in 'UnitList.pas';

const

dataFileName = 'list.data';

var

list: TList;

command: string;

itemTmp, itemTmp1, itemTmp2, itemTmp3: PData;

indexTmp: Integer;

n, i, Right: Integer;

begin

try

randomize;

list := TList.create(dataFileName);

repeat

writeln(#13#10 + '1 Item append');

writeln('2 Random remove');

writeln('3 Check: is this list correct to condi

tions.');

writeln('4 List output');

writeln('0 Exit');

write('Command: ');

n := (list.count+1) div 2;

readln(command);

if command = '1' then // append

begin

new(itemTmp);

Readln(itemTmp^.number);

list.appendItem(itemTmp);

writeln(itemTmp^.number,' is added');

end

else if command = '2' then // remove

if (list.count) = 0 then

writeln('List is empty')

else

begin

indexTmp := random(list.count);

writeln('Removing #', indexTmp+1);

list.removeItemByIndex(indexTmp);

end

else if command = '3' then // expression

if list.count = 0 then

writeln('List is empty')

else

begin

Right:=0;

for i := 0 to (n-1) do

begin

itemTmp1 := list[i];

itemTmp2 := list[n-1-i];

itemTmp3 := list[2\*n-1-i];

if itemTmp1^.number = (2\*itemTmp2^.number) +

itemTmp3^.number then

Inc(Right);

end;

if Right = n then

writeln(' Correct.')

else

Writeln(' Not Correct.');

end

else if command = '4' then // list output

list.consoleOutput()

else if command <> '0' then

writeln('Error');

until command = '0';

list.destroy;

except

on E: Exception do

writeln(E.ClassName, ': ', E.Message);

end;

end.

unit UnitList;

interface

type

TData = packed record

number: integer;

end;

PData = ^TData;

TListNode = class

private

item: PData;

next: TListNode;

prev: TListNode;

constructor create(const aItem: PData);

end;

TList = class

private

head: TListNode;

tail: TListNode;

fileName: string;

fCount: integer;

function getNodeByIndex(const i: integer): TListNode;

procedure removeNode(const node: TListNode);

public

constructor create(const aFileName: string); overload;

destructor destroy();

procedure appendItem(const item: PData);

procedure removeItemByIndex(const i: integer);

function getItemByIndex(const i: integer): PData;

property items[const i: integer]: PData read

getItemByIndex; default;

property count: integer read fCount;

procedure consoleOutput(); // äëÿ äåìîíñòðàöèè;

end;

implementation

uses

SysUtils;

constructor TListNode.create(const aItem: PData);

begin

inherited create();

item := aItem;

end;

constructor TList.create(const aFileName: string);

var

f: file of TData;

itemTmp: PData;

begin

create();

fileName := aFileName;

assignFile(f, fileName);

if not fileExists(filename) then

rewrite(f)

else

begin

reset(f);

while not eof(f) do

begin

new(itemTmp);

read(f, itemTmp^);

appendItem(itemTmp);

end;

end;

closeFile(f);

end;

destructor TList.destroy();

var

f: File of TData;

nodeCurr: TListNode;

begin

if fileName <> '' then

begin

assignFile(f, fileName);

rewrite(f);

nodeCurr := head;

while nodeCurr <> nil do

begin

write(f, nodeCurr.item^);

nodeCurr := nodeCurr.next;

end;

closeFile(f);

end;

inherited destroy();

end;

function TList.getNodeByIndex(const i: integer): TListNode;

var

j: integer;

begin

if i > fCount - 1 then

raise exception.create('Index out of range');

result := head;

for j := 1 to i do

result := result.next;

end;

procedure TList.removeNode(const node: TListNode);

begin

if node.prev <> nil then

begin

(node.prev).next := node.next;

if node.next = nil then

tail := node.prev

else

(node.next).prev := node.prev;

end

else

begin

head := node.next;

if head = nil then

tail := nil

else

(self.head).prev := nil

end;

dispose(node.item);

node.destroy();

dec(fCount);

end;

procedure TList.appendItem(const item: PData);

var

newNode: TListNode;

begin

newNode := TListNode.create(item);

if head = nil then

begin

head := newNode;

tail := newNode;

end

else

begin

tail.next := newNode;

newNode.prev := tail;

tail := newNode;

end;

inc(fCount);

end;

procedure TList.removeItemByIndex(const i: integer);

begin

removeNode(getNodeByIndex(i));

end;

function TList.getItemByIndex(const i: integer): PData;

begin

result := getNodeByIndex(i).item;

end;

procedure TList.consoleOutput();

var

nodeCurr: TListNode;

i: integer;

begin

nodeCurr := head;

i := 1;

while nodeCurr <> nil do

begin

writeln(i, ') ', nodeCurr.item^.number);

inc(i);

nodeCurr := nodeCurr.next;

end;

end;

end.

Приложение Б

(обязательное)

Тестовые наборы

Тест 1

Исходные данные:

1, 1, 1, -1, -1, -1

Ожидаемый результат:

Данная последовательность подходит условию.

Полученный результат: рисунок 6

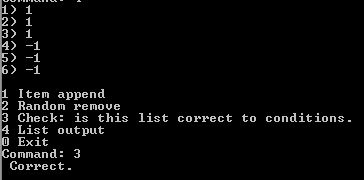


Рисунок 6 – Результат работы программы

Тест 2

Исходные данные:

1, 1, 1, -1, -1, 1

Ожидаемый результат:

Данная последовательность не подходит условию.

Полученный результат: рисунок 7

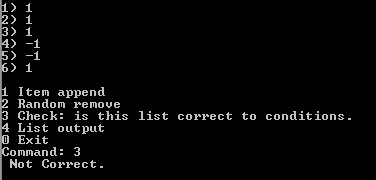


Рисунок 7 – Результат работы программы