**1. Условие задачи**

Описать собственный тип С1-связного линейного списка.

Создать и заполнить данными С1-связный линейный список, упорядоченный по С2, из N С3 С4 и найти элемент, удовлетворяющий условию С5. Вывести созданный список или сообщение: «Список пуст». Для двусвязных списков – созданный список выводить дважды: в прямом и обратном порядке.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **С1** | **С2** | **С3\*** | **С4** | **С5** |
| Дву- | По убыванию длины слов | Псевдослучайных, из массива слов | слов – названий цветов | Номер первого слова на букву Б, В или Г |

**2. Уточненное условие задачи**

Создать двусвязный список, элементы которого выполнены в виде записи, содержащие информацию вида: слова-названия цветов(или псевдослучайное слово) (строка), указатель на следующий элемент списка и на предыдущий, новый элемент добавлять в порядке убывания слова.

**TInfo** = string; //Тип, каким обладает информация в списке

**TStrSet** = array[0..60] of string; //Тип - массив строк, испольуется для словаря

**PElem** = ^TElem; //Указатель на элемент списка

**TElem** = record //Элемент списка

**info:** TInfo; //Состоит из информации, содеражащейся в ней

**next:** PElem; //Указатель на следующий

**prev:** PElem; //И на предыдущий элементы списка

end;

**Функциональные тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Исходные данные** | **Ожидаемый результат** | **Смысл теста** |
| 1 | Мать-и-мачеха  Василек  Фиалка  Васил | //ViewList  Прямой порядок:  Мать-и-мачеха,Василек,Фиалка,Васил  Обратный порядок:  Васил,Фиалка,Василек,Мать-и-мачеха  //SearchIt  Найдена строка, удовлетворяющая заданным условием. Её номер:2 | Список упорядочен по правилу |
| 2 | Васил  Фиалка  Василек  Мать-и-мачеха | //ViewList  Прямой порядок:  Мать-и-мачеха,Василек,Фиалка,Васил  Обратный порядок:  Васил,Фиалка,Василек,Мать-и-мачеха  //SearchIt  Найдена строка, удовлетворяющая заданным условием. Её номер:2 | Список упорядочен в обратном порядке |
| 3 | Ромашка  Фиалка  Роза  Василек  Одуванчик | //ViewList  Прямой порядок:  Одуванчик, Василек, Ромашка, Фиалка, Роза  Обратный порядок:  Роза, Фиалка, Ромашка, Василек, Одумашка  //SearchIt  Найдена строка, удовлетворяющая заданным условием. Её номер:2 | Неупорядоченный список |
| 4 | Василек | //ViewList  Прямой порядок:  Василек  Обратный порядок:  Василек  //SearchIt  Найдена строка, удовлетворяющая заданным условием. Её номер:1 | Один элемент списка |
| 5 | Ромашка  Фиалка  Роза  Красный  Одуванчик | //ViewList  Прямой порядок:  Одуванчик, Красный, Ромашка, Фиалка, Роза  Обратный порядок:  Роза, Фиалка, Ромашка, Красный, Одуванчик  //SearchIt  Нет таких элементов списка | Нет элемента, соответствующему правилу С5 |
| 6 | Янтарный  Белый  Фиалковый  Бирюзовый | //ViewList  Прямой порядок:  Бирюзовый, Фиалковый, Янтарный, Белый  Обратный порядок:  Белый, Янтарный, Фиалковый, Бирюзовый  //SearchIt  Найдена строка, удовлетворяющая заданным условием. Её номер:1 | Два элемента удовлетворяют правилу С5. Берется первый из них |
| 7 |  | //ViewList  Пустой  //SearchIt  Нет таких элементов списка | Пустой файл.  Соответствующее сообщение |

**program Lab3;**

{$APPTYPE CONSOLE}

uses

SysUtils,

Windows,

UnLab3 in 'UnLab3.pas',

UnVar in 'UnVar.pas',

UnFunc in 'UnFunc.pas';

var

ch: char;

begin

SetConsoleCP(1251); // из модуля Windows

SetConsoleOutputCP(1251);

Randomize; //PseudoRandom seed

repeat

writeln('--------------------------------------');

writeln('Q - добавить в список из текстового файла; ');

writeln('W - сгенерировать и добавить в список; ');

writeln('E - добавить в список; ');

writeln('R - найти в списке;');

writeln('T - удалить(очистить) список;');

writeln('Y - просмотр;');

writeln('U - конец.');

write('Ваш выбор?'); readln(ch);

writeln('--------------------------------------');

ch:= UpCase(ch);

case ch of

//----------создать список из текстового файла------------------

'Q': AddFromTextFile;

//----------сгенерировать список -------------------------------

'W': AddN;

//----------добавить в список ----------------------------------

'E': Add1;

//-----------поиск----------------------------------------------

'R': SearchIt;

//-----------освобождение памяти--------------------------------

'T': FreeList(NachaloSpiska, KonecSpiska);

//-----------просмотр----------------------------------------------

'Y': ViewList;

//-----------выход----------------------------------------------

'U': exit;

//--------------------------------------------------------------

else

begin

writeln('Нет такой команды');

write('Press ENTER'); readln;

end;

end;

until ch='U';

end.

**unit UnFunc**;

interface

uses

UnVar,

SysUtils,

UnLab3;

procedure AddR(var ListN, ListK: PElem; r: TInfo); // добавить элемент со значением R в соответствии с заданным порядком в список

procedure AddN; // "Добавить псевдослучайные N элементов" в текущий список

procedure AddFromTextFile; // Добавление всех данных из текстового файла в текущий список

procedure Add1; // Добавить один элемент в текущий список

procedure SearchIt; // "Найти в списке"

procedure ViewList; // "Что сейчас в списке?" - вывод на экран консоли

implementation

// добавить элемент со значением R в соответствии с заданным порядком в список

procedure AddR(var ListN, ListK: PElem; r: TInfo); // адреса начала и конца списка и R

var ListC: PElem; // текущий элемент списка

begin

if ListN = nil then CreateList(ListN, ListK, r) // если список пуст

else

if Length(r) >= Length(ListN^.info) then AddFirst(ListN, ListK, r) // добавить в начало

else

if Length(r) <= Length(ListK^.info) then AddLast(ListN, ListK, r) // добавить в конец

else

begin // найти после какого

ListC:=ListN; // с начала списка

while Length(ListC^.next^.info) >= Length(r) do

ListC := ListC^.next;

AddMedium(ListN, ListC, ListK, r) // добавить после текущего

end;

end;

// "Добавить псевдослучайные N элементов" в текущий список

procedure AddN;

var n,i: integer; r: Integer;

const words:array[0..14] of string = ('жил','был','бюрер','нетужил','пришел','дегтярев','бюрера','не','стало','бывает','же','такое','ОлегБылЗдесь','Всем','Пока');

begin

write('Сколько N=?'); readln(n);

for i:=0 to n-1 do

begin

r:=Random(14);

AddR(NachaloSpiska, KonecSpiska, words[r]);

end;

end;

// Добавление всех данных из текстового файла в текущий список

procedure AddFromTextFile;

var

f: TextFile;

r: TInfo;

begin

AssignFile(f, ParamStr(1));

try

Reset(f);

try

try

while not eof(f) do

begin

readln(f, r);

AddR(NachaloSpiska, KonecSpiska,r);

end;

writeln('Список создан');

except

writeln('Некорректные данные в текстовом файле '+ ParamStr(1));

end;

finally

CloseFile(f);

end;

except

writeln('Не удалось открыть текстовый файл '+ ParamStr(1));

end;

end;

// Добавить один элемент в текущий список

procedure Add1;

var

r: TInfo;

begin

try

write('введите название цветка R:'); readln(r);

AddR(NachaloSpiska, KonecSpiska, r);

except

writeln('Некорректное число или ошибка добавления');

end;

end;

// "Найти в списке"

procedure SearchIt;

var

ListC: PElem;

isFound:Boolean;

const letters:set of char = ['Б','В','Г'];

begin

ListC:=NachaloSpiska;

isFound:=false;

while (ListC<>nil) and (not isFound) do

begin

if ((AnsiUpperCase(ListC^.info))[1] in letters) then

begin

Writeln('Найдена строка, начинающаяся на одну из заданных букв:',ListC^.info);

isFound:=True;

end

else ListC:=ListC^.next;

end;

if not isFound then writeln('Нет таких элементов списка');

end;

// "Что сейчас в списке?" - вывод на экран консоли

procedure ViewList;

var

ListC: PElem;

begin

writeln('Список:');

ListC:=NachaloSpiska;

if ListC=nil then writeln('Пустой')

else

begin

Writeln('Прямой порядок:');

while ListC<>nil do

begin

writeln(ListC^.info);

ListC:=ListC^.next;

end;

Writeln('Обратный порядок:');

ListC:=KonecSpiska;

while ListC<>nil do

begin

writeln(ListC^.info);

ListC:=ListC^.prev;

end;

end;

end;

end.

unit **UnLab3**;

interface

type

TInfo = string;

PElem = ^TElem;

TElem = record

info: TInfo;

next: PElem;

prev: PElem;

end;

// создать список из одного элемента

procedure CreateList(var ListN, ListK: PElem; r: TInfo);

// добавить новый элемент перед первым

procedure AddFirst(var ListN: PElem; ListK: PElem; r: TInfo);

// добавить новый элемент в конец списка

procedure AddLast(ListN: PElem; var ListK: PElem; r: TInfo);

// добавить новый элемент в середину после ListC (не в конец)

procedure AddMedium(ListN,ListC,ListK: PElem; r: TInfo);

// очистить список

procedure FreeList(var ListN, ListK: PElem);

//получить слова

procedure GetDict(var Dict:TStrSet; var DictLen:byte);

//Проверка на наличие слова в словаре

function inDict(var Dict:TStrSet; var DictLen:byte;const r:TInfo):boolean;

//записать получившийся словарь

procedure WriteDict(const Dict:TstrSet; const DictLen:byte);

implementation

// создать список из одного элемента

procedure CreateList(var ListN, ListK: PElem; r: TInfo);

begin

New(ListN);

ListK:=ListN; // конец и начало совпадают

ListN^.info:=r;

ListN^.next:=nil;

ListN^.prev:=nil;

end;

// добавить новый элемент перед первым

procedure AddFirst(var ListN: PElem; ListK: PElem; r: TInfo);

var Elem: PElem;

begin

new(Elem);

Elem^.info:=r;

Elem^.next:=ListN;

ELem^.next^.prev:=Elem;

Elem^.prev:=nil;

ListN:=Elem; // теперь он 1-ый

end;

// добавить новый элемент в конец

procedure AddLast(ListN: PElem; var ListK: PElem; r: TInfo);

begin

new(ListK^.next);

ListK^.next^.info:=r;

ListK^.next^.next:=nil;

ListK^.next^.prev:=ListK;

ListK:= ListK^.next;

end;

// добавить новый элемент в середину после ListC (не в конец)

procedure AddMedium(ListN,ListC,ListK: PElem; r: TInfo);

var Elem: PElem;

begin

new(Elem);

Elem^.info:=r;

Elem^.next:=ListC^.next; // после него то, что было после ListC

Elem^.next^.prev:=Elem;

Elem^.prev:=ListC;

ListC^.next:=Elem; // а он сам после ListC

end;

// очистить список

procedure FreeList(var ListN, ListK: PElem);

var Elem: PElem;

begin

Elem:=ListN;

while Elem<>nil do

begin

ListN:=ListN^.next;

Dispose(Elem);

Elem:=ListN;

end;

ListK:=nil;

end;

procedure GetDict;

var inp:TextFile; i:Integer; buff:string;

begin

AssignFile(inp,'in/dict.txt');

Reset(inp);

try

i:=0;

while not(eof(inp)) do

begin

Readln(inp,buff);

Dict[i]:=buff;

Inc(i);

end;

DictLen:=i;

finally CloseFile(inp); end;

end;

function inDict;

var i:Byte;

begin

result:=False;

for i:=0 to DictLen-1 do

if Dict[i]=r then

begin

result:=True;

exit;

end;

end;

procedure WriteDict;

var outp:TextFile; i:Integer;

begin

AssignFile(outp,'in/dict.txt');

Rewrite(outp);

for i:=0 to DictLen-1 do

begin

Writeln(outp,Dict[i]);

end;

CLoseFile(outp);

end;

end.

**unit UnVar**;

interface

uses UnLab3; // описание типа и все процедуры

var NachaloSpiska, KonecSpiska: PElem; // начало и конец списка

implementation

// пусто

initialization

NachaloSpiska:=nil; // инициализация переменных

KonecSpiska:=nil;

finalization

FreeList(NachaloSpiska, KonecSpiska); // освобождение памяти

end.