Лабораторная работа № 6

по дисциплине “Алгоритмы и структуры данных”

Тема: Динамические переменные и линейные структуры данных.

## Линейные списки.

**Цель работы:**

Получить практические навыки работы с динамическими переменными и динамическими структурами данных.

## Общие сведения

Для работы с динамическими структурами данных используются указатели. Указатели представляют собой специальный тип данных. Они принимают значения, равные адресам размещения в оперативной памяти соответствующих динамических переменных.

Списком называется структура данных, каждый элемент которой посредством указателя связывается со следующим элементом. На самый первый элемент (голову списка) имеется отдельный указатель.

Из определения следует, что каждый элемент списка содержит поле данных (оно может иметь сложную структуру) и поле ссылки на следующий элемент. После ссылки последнего элемента должно содержать пустой указатель (nil).

Число элементов связанного списка может расти или уменьшаться в зависимости от того, сколько данных мы хотим хранить в нем. Чтобы добавить новый элемент в список, необходимо:

1. Получить память для него;

2. Поместить туда информацию;

3. Добавить элемент в конец списка (или начало).

Элемент списка состоит из разнотипных частей (хранимая информация и указатель), и его естественно представить записью. Перед описанием самой записи описывают указатель на нее:

Type { описание списка из целых чисел }

PList = ^TList;

TList = record

Inf : Integer;

Next : PList;

end;

## Примеры

**Создание списка.**

*Задача.* Сформировать список, содержащий целые числа 3, 5, 1, 9.

Определим запись типа TList с полями, содержащими характеристики данных – значения очередного элемента и адреса следующего за ним элемента

PList = ^TList;

TList = record

Data : Integer;

Next : PList;

end;

Чтобы список существовал, надо определить указатель на его начало. Опишем переменные.

Var

Head, x : PList;

Создадим первый элемент: New(Head); { выделяем место в памяти для переменной Head } Head^.Next := nil; { указатель на следующий элемент пуст (такого элемента нет) } Head^.Data := 3; { заполняем информационное поле первого элемента }

image62

Продолжим формирование списка, для этого нужно добавить элемент в конец списка.

Введем вспомогательную переменную указательного типа, которая будет хранить адрес последнего элемента списка: x := Head; {сейчас последний элемент списка совпадает с его началом}

image64

New(x^.Next); { выделим области памяти для следующего (2-го) элемента и поместим его адрес в адресную часть предыдущего (1-го) элемента }

image66

x := x^.Next ; { переменная x принимает значение адреса выделенной области. Таким образом осуществляется переход к следующему (2-ому) элементу списка }

image68

x^.Data := 5; { значение этого элемента } x^.Next := nil; {следующего значения нет }

image70

Остальные числа заносятся аналогично:

New(x^.Next); { выделим области памяти для следующего элемента }

x := x^.Next ; { переход к следующему (3-му) элементу списка }

x^.Data := 1; { значение этого элемента }

x^.Next := nil; {следующего значения нет }

New(x^.Next); { выделим области памяти для следующего элемента }

x := x^.Next ; { переход к следующему (4-му) элементу списка }

x^.Data := 9; { значение этого элемента }

x^.Next := nil; {следующего значения нет }

***Замечание.*** Как видно из примера, отличным является только создание первого (Head) элемента – головы списка. Все остальные действия полностью аналогичны и их естественно выполнять в цикле.

Присоединение нового элемента к голове списка производится аналогично:

………………..

New(x);

{ ввод значения элемента x^.Data := … }

x^.Next := Head;

Head := x;

………………..

В этом случае последний введенный элемент окажется в списке первым, а первый – последним.

**Просмотр списка.**

Просмотр элементов списка осуществляется последовательно, начиная с его начала. Указатель List последовательно ссылается на первый, второй и т. д. элементы списка до тех пор, пока весь список не будет пройден. При этом с каждым элементом списка выполняется некоторая операция– например, печать элемента. Начальное значение List – адрес первого элемента списка (Head). Digit – значение удаляемого элемента.

List := Head;

[While](file:///F:\delphi\help\delphi.asp?name=while) List^.Next <> nil do

begin

[WriteLn](file:///F:\delphi\help\delphi.asp?name=writeln)(List^.Data);

List := List^.Next; { переход к следующему элементу; аналог для массива i:=i+1 }

end;

**Удаление элемента из списка.**

При удалении элемента из списка необходимо различать три случая:

1. Удаление элемента из начала списка.

2. Удаление элемента из середины списка.

3. Удаление из конца списка.

*Удаление элемента из начала списка.*

List := Head; { запомним адрес первого элемента списка }

Head := Head^.List; { теперь Head указывает на второй элемент списка }

Dispose(List); { освободим память, занятую переменной List^ }

*Удаление элемента из середины списка.*

Для этого нужно знать адреса удаляемого элемента и элемента, находящегося в списке перед ним.

List := Head;

[While](file:///F:\delphi\help\delphi.asp?name=while) (List<>nil) and (List^.Data<>Digit) do

begin

x := List;

List := List^.Next;

end;

x^.Next := List^.Next;

Dispose(List);

Удаление из конца списка.

Оно производится, когда указатель х показывает на предпоследний элемент списка, а List – на последний.

List := Head; x := Head;

[While](file:///F:\delphi\help\delphi.asp?name=while) List^.Next<>nil do

begin

x := List;

List := List^.Next;

end;

x^.Next := nil;

Dispose(List);

## 2. Стек

**Цель работы:**

Закрепить технику работы с динамическими структурами данных на примере стека. Рассмотреть основные операции со стеком и познакомится с типичными примерами применения стека.

## Общие сведения

Стек – упорядоченный набор элементов, в котором добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека.

В любой момент времени доступен лишь один элемент стека – верхний. Извлекать элементы из стека можно только в порядке, обратном их добавления в стек (первым пришел, последним ушел). Значением указателя, представляющего стек, является ссылка на вершину стека, каждый элемент стека содержит поле ссылки на следующий элемент.

Таким образом, описать стек, содержащий, например, целые числа, можно следующим образом:

Type

PStack = ^TStack;

TStack = Record

Data : Integer;

Next : PStack;

end;

Var

Stack : PStack;

Если стек пуст, то значение переменной Stack равно nil.

## Примеры

**Занесение в стек.**

Занесение в стек производится аналогично вставке нового элемента в начало списка:

Var

x : PStack;

……………

[New](file:///F:\delphi\help\delphi.asp?name=new)(x);

x^.Data := …….;

x^.Next := Stack;

Stack := x;

……………

**Извлечение элемента из стека.**

В результате выполнения этой операции некоторой переменной N должно быть присвоено значение первого элемента стека и изменено значение указателя на начало стека:

Var

N : Integer;

x : PStack;

………………….

N := Stack^.Data;

x:= Stack;

Stack := Stack^.Next;

Dispose(x);

………………….

## 1. Контрольные вопросы

1. Что такое указатели? Какие значения они могут принимать? Какие операции возможны над указателями?

2. Что представляют собой динамические структуры данных? Для чего они используются? Чем отличаются от данных статического типа?

3. Какие стандартные процедуры существуют для работы с указателями?

4. Зачем различать типы указателей?

5. Какие операции требуется выполнить для вставки и удаления элемента списка?

6. Сколько элементов может содержать список?

7. Можно ли для построения списка обойтись одной переменной?

## 1.Варианты заданий

## Для представления исходных и выходных данных используйте необходимые компоненты вместо текстовых файлов.

**1.** Сформировать список строк и а) сохранить его в текстовом файле; б) сохранить его в обратном порядке в текстовом файле. Использовать рекурсию.

**2.** Сформировать список строк из текстового файла и подсчитать количество символов во всех элементах списка.

**3.** Написать функцию, которая вычисляет среднее арифметическое элементов непустого списка.

**4.** Написать процедуру присоединения списка L2 к списку L1.

**5.** Написать функцию, которая создает список L2, являющийся копией списка L1, начинающегося с данного узла.

**6.** Написать функцию, которая подсчитывает количество вхождений ключа в списке.

**7.** Написать функцию, которая удаляет из списка все вхождения ключа.

**8.** Многочлен image72задан своими коэффициентами, которые хранятся в форме списка. Написать функции:

– Equal(p, q), проверяющую на равенство многочлены p и q;

– Summa(p, q, r), которая строит многочлен r = p + q.

**9.** Вычислить значение многочленаimage74 в целочисленной точке x. Коэффициенты вводятся с клавиатуры и динамически размещаются в памяти.

**10.** Сформировать список целых чисел и упорядочить их по неубыванию.

**11.** Сформировать список целых чисел и удалить из него все четные.

**12.** Сформировать список вещественных чисел и вычислить сумму.

**13.** Написать рекурсивную и нерекурсивную процедуры проверки наличия в списке заданного числа.

**14.** Написать функцию, которая проверяет, упорядочены ли элементы списка по алфавиту.

**15.** Написать функцию, подсчитывающую количество слов в списке, которые начинаются с той же буквы, что и следующее слово.

**16.** Определить симметричность произвольного текста любой длины. Текст должен оканчиваться точкой. Задачу решить с помощью двух списков.

**17.** Вычислить значение выраженияimage76. Значенияimage78 вводятся с клавиатуры и динамически размещаются в памяти.

**18.** Написать функцию, которая использует исходный список L и создает два новых списка L1 и L2. L1 содержит нечетные узлы, а L2 – четные.

**19.** Написать функцию, которая использует исходный список L и создает два новых списка L1 и L2. L1 содержит нечетные числа, а L2 – четные.

**20.** Сформировать два списка, отсортировать их объединить в один, не нарушая порядка.

## 2.Контрольные вопросы

1. В чем сходство и отличие динамических структур данных типа список и стек?

2. Можно ли добраться до середины или конца («дна») стека, минуя его начало («вершину»)?

3. Приведите примеры из жизни, где встречается «принцип стека».

4. Оформите в виде процедуры Push занесение в стек значения.

5. Оформите в виде процедуры Pop извлечение значения из стека.

## 2.Варианты заданий

**1.** Подсчитать количество элементов в стеке.

**2.** Сформировать стек, содержащий строки и сохранить его в текстовом файле.

**3.** Восстановить стек, содержащий строки, из текстового файла.

**4.** Написать функцию, которая вычисляет среднее арифметическое элементов стека.

**5.** Написать процедуру присоединения стека S2 к стеку S1.

**6.** Определить симметричность произвольного текста любой длины. Текст должен оканчиваться точкой. Задачу решить с помощью двух стеков.

**7.** Слить два стека, содержащих возрастающую последовательность целых положительных чисел, в третий стек так, чтобы его элементы располагались также в порядке возрастания.

**8.** В данном тексте проверить соответствие открытия и закрытия скобок.

**9.** Напечатать содержимое текстового файла, выписывая символы каждой его строки в обратном порядке.

**10.** Проверить, является ли строка палиндромом.