**Частное учреждение образования**

**«Гродненский колледж бизнеса и права»**

**Лабораторная работа № 1**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Динамические структуры данных

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 1**

Тема: Динамические структуры данных.

Цель: Сформировать навыки программирования динамических структур данных.

Задачи: Освоение теоретического материала, выполнение индивидуальной работы.

**ЗАДАЧИ**

Условие: Натуральное число из n цифр является числом Армстронга, если сумма его цифр, возведенных в n-ую степень равна самому числу, например, 153=13+53+33. Получить все числа Армстронга, состоящие из двух, трех и четырех цифр, организовать соответственно 3 однонаправленных списка.

Алгоритм: Предоставлен в письменном виде преподавателю.

Решение: Использован модуль «Niger1» к основной программе «Est1».

Модуль:

**unit** Niger1;

**interface**

**type**

{Тип основных данных.}

TData = Integer;

{Тип, описывающий указатель на элемент.}

TPElem = ^TElem;

{Тип, описывающий элемент списка.}

TElem = **record**

Data : TData; {Основные данные элемента.}

PNext : TPElem; {Указатель на следующий элемент.}

**end**;

{Тип, описывающий список.}

TDList = **record**

PFirst, PLast : TPElem; {Указатели на первый и на последний элементы списка.}

**end**;

**procedure** Init(**var** aL : TDList);

**procedure** Add(**var** aL : TDList; **const** aData : TData);

**procedure** LWriteln(**const** aL : TDList);

**procedure** LFree(**var** aL : TDList);

**implementation**

**procedure** Init(**var** aL : TDList);

**begin**

aL.PFirst := nil;

aL.PLast := nil;

**end**;

**procedure** Add(**var** aL : TDList; **const** aData : TData);

**var**

P : TPElem;

**begin**

**New**(P); {Выделяем паямять для элемента.}

P^.Data := aData; {Записываем данные.}

P^.PNext := nil; {Обнуляем указатель на следующий элемент.}

**if** aL.PFirst = nil **then** {Если список пуст, то новый элемент становится первым элементом списка.}

aL.PFirst := P

**else** {Если список непустой, то новый элемент прикрепляем к последнему элементу списка.}

aL.PLast^.PNext := P;

aL.PLast := P; {Новый элемент становится последним элементом списка.}

**end**;

**procedure** LWriteln(**const** aL : TDList);

**var**

P : TPElem;

**begin**

P := aL.PFirst; {Указатель на первый элемент списка.}

**if** P <> nil **then**

**repeat**

**if** P <> aL.PFirst **then** {Если элемент не первый в списке, то ставим перед ним запятую.}

Write(', ');

Write(P^.Data); {Распечатываем данные текущего элемента.}

P := P^.PNext; {Получаем указатель на следующий элемент.}

**until** P = nil

**else**

Write('Список пуст.');

Writeln;

**end**;

**procedure** LFree(**var** aL : TDList);

**var**

P, PDel : TPElem;

**begin**

P := aL.PFirst; {Указатель на первый элемент списка.}

**while** P <> nil **do**

**begin**

PDel := P; {Запоминаем указатель на текущий элемент.}

P := P^.PNext; {Получаем указатель на следующий элемент.}

Dispose(PDel); {Освобождаем память, занятую текущим элементом списка.}

**end**;

Init(aL);

**end**;

**end**.

Программа:

**uses** Niger1;

**var**

L1, L2, L3: TDList;

Data: TData;

i, p, a, b, c, d, arm: Integer;

S: String;

**begin**

Init(L1); {Начальная инициализация списка.}

{Создаём список.}

**for** i := 10 **to** 99 **do**

**begin**

p := i;

arm := 0;

**begin**

a := p **mod** 10;

b := p **div** 10;

arm := sqr(a) + sqr(b);

**end**;

**if** arm = i **then**

**begin**

Data := i;

Add(L1, Data);

**end**;

**end**;

Writeln('Задан список 1:');

LWriteln(L1);

Init(L2); {Начальная инициализация списка.}

{Создаём список.}

**for** i := 100 **to** 999 **do**

**begin**

p := i;

arm := 0;

**begin**

a := p **mod** 10;

b := p **div** 10 **mod** 10;

c := p **div** 100;

arm := c \* c \* c + b \* b \* b + a \* a \* a;

**end**;

**if** arm = i **then**

**begin**

Data := i;

Add(L2, Data);

**end**;

**end**;

Writeln('Задан список 2:');

LWriteln(L2);

Init(L3); {Начальная инициализация списка.}

{Создаём список.}

**for** i := 1000 **to** 9999 **do**

**begin**

p := i;

arm := 0;

**begin**

a := p **mod** 10;

b := p **div** 10 **mod** 10;

c := p **div** 1000;

d := p **div** 100 **mod** 10;

arm := c \* c \* c \* c + b \* b \* b \* b + a \* a \* a \* a + d \* d \* d \* d;

**end**;

**if** arm = i **then**

**begin**

Data := i;

Add(L3, Data);

**end**;

**end**;

Writeln('Задан список 3:');

LWriteln(L3);

**end**.

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Каково назначение переменных ссылочного типа?

Указатель – это переменная, которая в качестве своего значения содержит адрес байта памяти.

2. Как распределяется память под переменные ссылочного типа?

Значения указателей – адреса переменных в памяти. Адрес занимает четыре байта и хранится в виде двух слов, одно из которых определяет сегмент, второе – смещение.

3. Каково назначение процедур NEW и DISPOSE, MARK и RELEASE?

NEW выделяет память под динамическую переменную.

DISPOSE освобождает динамическую память.

MARK Записывает состояние кучи в указатель.

RELEASE Возвращает кучу в заданное состояние.

4.

5.Дайте определение динамической переменной?

Динамическая переменная - переменная в программе, место в оперативной памяти под которую выделяется во время выполнения программы.

6.Чем отличается динамическая переменная от статической?

Для статической переменной память выделяется при компиляции программы(запуске), а для динамической в любой момент исполнения программы.

7.Что понимается в языке Паскаль под кучей?

Вся динамическая память, в виде сплошного массива байтов.

8.Какие операции выполняются над переменными ссылочного типа?

В Паскале определены стандартные функции для работы с указателями:

* addr( x) – тип результата pointer, возвращает адрес x (аналогично операции @), где x – имя переменной или подпрограммы;
* seg( x) – тип результата word, возвращает адрес сегмента для x;
* ofs( x) – тип результата word, возвращает смещение для x;
* ptr( seg, ofs) – тип результата pointer, по заданному сегменту и смещению формирует адрес типа pointer.

9.Как организуется однонаправленный список ?

Каждый элемент связанного списка, во-первых, хранит какую-либо информацию, во-вторых, указывает на следующий за ним элемент. Так как элемент списка хранит разнотипные части (хранимая информация и указатель), то его естественно представить записью, в которой в одном поле располагается объект, а в другом – указатель на следующую запись такого же типа. Такая запись называется звеном, а структура из таких записей называется списком или цепочкой.

Лишь на самый первый элемент списка (голову) имеется отдельный указатель. Последний элемент списка никуда не указывает.

10. Каким образом можно исключить элемент из списка?

А)Удаление первого элемента. Для этого во вспомогательном указателе запомним первый элемент, указатель на голову списка переключим на следующий элемент списка и освободим область динамической памяти, на которую указывает вспомогательный указатель.

Б) Удаление элемента из середины списка. Для этого нужно знать адреса удаляемого элемента и элемента, стоящего перед ним. Допустим, что digit – это значение удаляемого элемента.

В)Удаление из конца списка. Для этого нужно найти предпоследний элемент.

11.Как организуется вставка элемента в список?

А) Добавим элемент в голову списка. Для этого необходимо выполнить последовательность действий:

* получить память для нового элемента;
* поместить туда информацию;
* присоединить элемент к голове списка.

12.Каким образом можно добавить элемент в конец списка?

Б)Добавление элемента в конец списка. Для этого введем вспомогательную переменную, которая будет хранить адрес последнего элемента. Пусть это будет указатель с именем hv (хвост).