ЛАБОРАТОРНАЯ РАБОТА
«Динамические структуры данных»

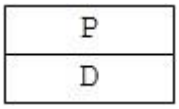
Цель работы: Изучение основных приемов использования указателей для создания динамических структур данных, отображаемых на списки.

Узел списка представляется в программе структурой, поля которой отражают его содержательную и управляющую части. Содержательная часть определяется особенностями решаемой задачи, а управляющая часть – это указатели на предыдущий и следующий узлы. Представлением для списка в целом является указатель на первый элемент списка. Для пустого списка – пустой указатель (NULL).

Выделяются следующие динамические структуры данных:

- однонаправленные (односвязные)
- списки; двунаправленные (двусвязные)
- списки; циклические списки; стек; дек;
- очередь; бинарные деревья.
-
- **Объявление динамических структур данных**
-

Каждая компонента любой динамической структуры представляет собой запись, содержащую, по крайней мере, два поля: одно поле типа указатель, а второе – для размещения данных. В общем случае запись может содержать не один, а несколько указателей и несколько полей данных. Поле данных может быть переменной, массивом или структурой. Для наилучшего представления изобразим отдельную компоненту в виде:



где поле P – указатель; поле D – данные.

Элемент динамической структуры состоит из двух полей:

- информационного поля (поля данных), в котором содержатся те данные, ради которых и создается структура; в общем случае информационное поле само является интегрированной структурой – вектором, массивом, другой динамической структурой и т.п.;
- адресного поля (поля связок), в котором содержатся один или несколько указателей, связывающий данный элемент с другими элементами структуры;

Объявление элемента динамической структуры данных выглядит следующим образом:
struct имя_типа { информационное поле; адресное поле; };
Например:

```
struct TNode {
        int Data;        //информационное поле
TNode *Next;  //адресное поле  };
```

Информационных и адресных полей может быть как одно, так и несколько.

Рассмотрим в качестве примера динамическую структуру, схематично указанную на рис.

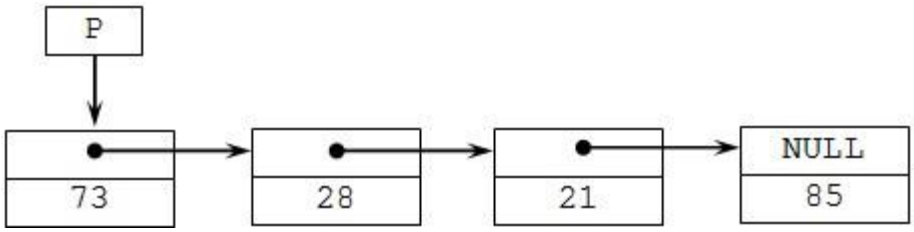


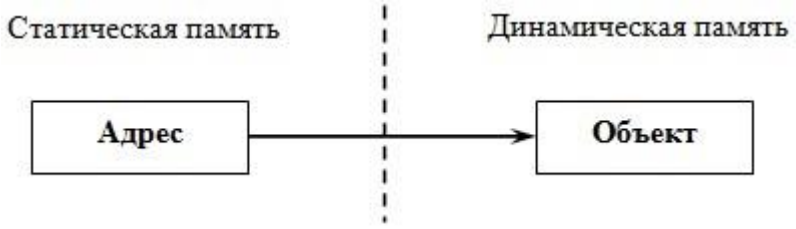
Рис. Схематичное представление динамической структуры

Данная структура состоит из 4 элементов. Ее первый элемент имеет поле Data, равное 73, и связан с помощью своего поля Next со вторым элементом, поле Data которого равно 28, и так далее до последнего, четвертого элемента, поле Data которого равно 85, а поле Next равно NULL, то есть нулевому адресу, что является признаком завершения структуры. Здесь P является указателем, который указывает на первый элемент структуры.

Доступ к данным в динамических структурах

Элемент динамической структуры в каждый момент может либо существовать, либо отсутствовать в памяти, поэтому его называют динамическим. Поскольку элементами динамической структуры являются динамические переменные, то единственным средством доступа к динамическим структурам и их элементам является указатель (адрес) на место их текущего расположения в памяти. Таким образом, доступ к динамическим данным выполняется специальным образом с помощью указателей.

Указатель содержит адрес определенного объекта в динамической памяти. Адрес формируется из двух слов: адрес сегмента и смещение. Сам указатель является статическим объектом и расположен в сегменте данных.



Для обращения к динамической структуре достаточно хранить в памяти адрес первого элемента структуры. Поскольку каждый элемент динамической структуры хранит адрес следующего за ним элемента, можно, двигаясь от начального элемента по адресам, получить доступ к любому элементу данной структуры.

Доступ к данным в динамических структурах осуществляется с помощью операции "стрелка" (->), которую называют операцией косвенного выбора элемента структурного объекта, адресуемого указателем. Она обеспечивает доступ к элементу структуры через адресующий ее указатель того же структурного типа.

Формат применения данной операции следующий: **УказательНаСтруктуру-> ИмяЭлемента** Операции "стрелка" (->) двуместная. Применяется для доступа к элементу, задаваемому правым операндом, той структуры, которую адресует левый операнд. В качестве левого операнда должен быть указатель на структуру, а в качестве правого – имя элемента этой структуры.

Например: p->Data;
p->Next;

Работа с памятью при использовании динамических структур

В программах, в которых необходимо использовать динамические структуры данных, работа с памятью происходит стандартным образом.

Выделение динамической памяти производится с помощью операции **new** или с помощью библиотечной функции **malloc (calloc).**

Освобождение динамической памяти осуществляется операцией **delete** или функцией **free**.

Пример

```
struct Node {char *Name;
          int          Value;
Node *Next
          };
Node *PNode; //объявляется указатель

PNode = new Node; //выделяется память

PNode->Name = "STO"; //присваиваются значения
PNode->Value = 28;
PNode->Next = NULL;

delete PNode; // освобождение памяти
```

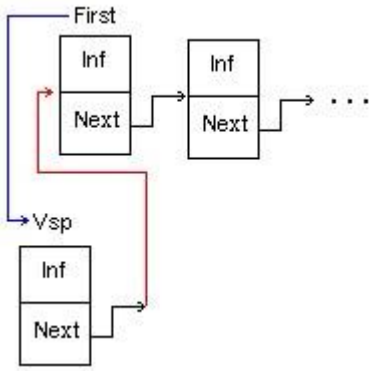
Примеры

На Паскале	**На Си**

Type U = ^Zveno;
Zveno = Record
Inf : BT;
Next: U
End;
	struct Zveno {
	BT Inf;
	Zveno * Next
	};

Здесь BT — некоторый базовый тип элементов списка.

1. Добавление звена в начало списка



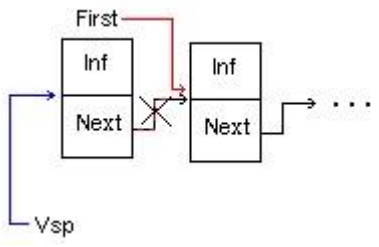
На Паскале

```
Procedure V_Nachalo(Var First : U; X : BT);
    Var Vsp : U;
    Begin
        New(Vsp);
        Vsp^.Inf := X;
        Vsp^.Next := First; {То звено, что было заглавным, становится вторым по счѐту}
        First := Vsp; {Новое звено становится заглавным}
End;
```

На Си

```
Zveno* Nachalo(Zveno* First, BT X)
{
   Zveno* Vsp = new Zveno;
   Vsp -> Inf = X;
   Vsp->Next=First;
First=Vsp;     return
First;
}
```

2. Удаление звена из начала списка



На Паскале

```
Procedure Iz_Nachala(Var First : U; Var X : BT);
    Var Vsp : U;
    Begin
        Vsp := First; {Забираем ссылку на текущее заглавное звено}
        First := First^.Next; {То звено, что было вторым по счѐту, становится заглавным}
        X := Vsp^.Inf; {Забираем информацию из удаляемого   звена}
        Dispose(Vsp); {Уничтожаем звено}
   End;
```

На Си

```
Zveno* Iz_Nachala(Zveno* First, BT X)
{
    Zveno* Vsp;
    Vsp = First;
    First = First->Next;
X = Vsp->Inf;
delete Vsp;       return
First;
}
```

ЗАДАНИЕ

Не создавая и не используя классы

1. Разработайте динамическую структуру «**стек»** для хранения целых значений. Используйте эту структуру данных для решения следующей задачи (можно использовать только те операции, которые определены для АТД «Стек» !!!).
   Последовательность чисел Фибоначчи задается по закономерности: f0 = 1, f1 = 1, ..., fn = fn-1 + fn-2. Распечатайте n чисел Фибоначчи в следующем порядке: сначала все четные, затем все нечетные элементы. Все получаемые числа Фибоначчи хранятся в стеке.

2. Создайте функции и программу-тест, реализующие основные операции для очереди, представленной двунаправленным списком. Узлы списка содержат целочисленные значения.

3. Реализовать представление и основные операции для множеств. Помимо добавления, удаления, проверки на принадлежность (и т.д.) элемента множества, реализовать операции объединения и пересечения множеств.

4. Организовать линейный список цветов: хранить название цвета и его числовой код. Данные в списке должны быть упорядочены по названию цвета.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличаются статические и динамические величины?
2. Почему для обращения к динамической структуре достаточно хранить в памяти адрес ее первого элемента?
3. За счет чего работа с динамическими данными замедляет выполнение программы?
4. Какого типа может быть поле данных в динамической структуре?
5. Как в программе представить узел списка?
6. Как представить список в целом? Пустой список?
7. Как через указатель на узел получить доступ к содержимому узла?
8. Какую дисциплину добавления/удаления элементов поддерживает динамическая структура "стек"?
9. Какую дисциплину добавления/удаления элементов поддерживает динамическая структура "очередь"?
10. Удобно ли реализовать динамическую структуру "стек" ("очередь") на основе массива? Списка? Двоичного дерева?
11. Даны два множества {-1, 0, 1} и { 1, 2, 3}. Каким будет их пересечение? Объединение?
12. Сколько аргументов (и какого типа) должно быть у функции, реализующей алгоритм нахождения пересечения двух множеств, если тип возвращаемого ею значения void?
13. Элементы множества хранятся в упорядоченном виде. Какой алгоритм поиска элемента множества Вы можете предложить?
14. Какие динамические структуры данных, кроме указанных в этой лабораторной работе, Вы знаете?