Лабораторная работа №5.

Динамические структуры данных.

Цель работы: изучение методов гибкой динамической организации данных.

Теоретические сведения

Важной особенностью любого языка программирования является его удобство при работе с *динамическими структурами данных*. Динамическими структурами (ДС) мы будем называть контейнеры данных, которые могут изменять свой размер в процессе выполнения программы. ДС позволяют добавлять в себя новые объекты путем выделения для них памяти, а также удалять объекты, освобождая память.

Если до начала работы с данными невозможно определить, сколько памяти потребуется для их хранения, память следует распределять во время выполнения программы по мере необходимости отдельными блоками. Блоки связываются друг с другом с помощью указателей.

ДС характеризуется тем что:

- не имеет имени;
- ей выделяется память в процессе выполнения программы;
- количество элементов структуры может не фиксироваться;
- размерность структуры может меняться в процессе выполнения программы;
- в процессе выполнения программы может меняться характер взаимосвязи между элементами структуры.

Каждой динамической структуре данных сопоставляется статическая переменная типа указатель (ее значение – адрес этого объекта), посредством которой осуществляется доступ к динамической структуре. Сами динамические величины не требуют описания в программе, поскольку во время компиляции память под них не выделяется. Во время компиляции память выделяется только под статические величины. Указатели – это статические величины, поэтому они требуют описания.

Динамические структуры, по определению, характеризуются отсутствием физической смежности элементов структуры в памяти, непостоянством и непредсказуемостью размера (числа элементов) структуры в процессе ее обработки. Поскольку элементы ДС располагаются по непредсказуемым адресам памяти, адрес элемента такой структуры не может быть вычислен из адреса начального или

предыдущего элемента. Для установления связи между элементами динамической структуры используются указатели, через которые устанавливаются явные связи между элементами. Такое представление данных в памяти называется связным.

Достоинства связного представления данных – в возможности обеспечения значительной изменчивости структур:

- размер структуры ограничивается только доступным объемом машинной памяти;
- при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;
- большая гибкость структуры.

Вместе с тем, связное представление не лишено и недостатков, основными из которых являются следующие:

- на поля, содержащие указатели для связывания элементов друг с другом, расходуется дополнительная память;
- доступ к элементам связной структуры может быть менее эффективным по времени.

Последний недостаток является наиболее серьезным и именно им ограничивается применимость связного представления данных. Если в смежном представлении данных для вычисления адреса любого элемента нам во всех случаях достаточно было номера элемента и информации, содержащейся в дескрипторе структуры, то для связного представления адрес элемента не может быть вычислен из исходных данных. Дескриптор связной структуры содержит один или несколько указателей, позволяющих войти в структуру, далее поиск требуемого элемента выполняется следованием по цепочке указателей от элемента к элементу. Поэтому связное представление практически никогда не применяется в задачах, где логическая структура данных имеет вид вектора или массива (с доступом по номеру элемента), но часто применяется в задачах, где логическая структура требует другой исходной информации доступа (таблицы, списки, деревья и т.д.).

Порядок работы с ДС следующий:

1. создать (отвести место в динамической памяти);
2. работать при помощи указателя;
3. удалить (освободить занятое структурой место).

Среди динамических структур данных выделяют следующие типы

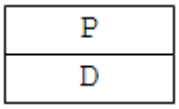
- однонаправленные (односвязные) списки;

- двунаправленные (двусвязные) списки;
- циклические списки;
- стек;
- дек;
- очередь;
- бинарные деревья.

Эти типы отличаются способом связи отдельных элементов и/или допустимыми операциями.

Объявление ДС

Каждый элемент ДС представляет собой запись, содержащую, по крайней мере, два поля: одно хранит указатель на следующий элемент, второе служит 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.

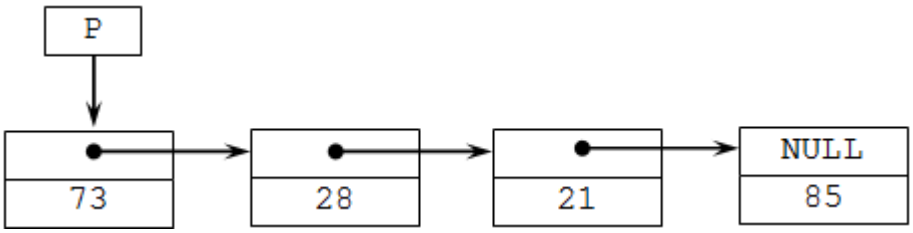
В языке C++ элемент динамической структуры данных может быть объявлен следующим образом:

```
struct имя_типа
{
    информационное поле;
    адресное поле;
};
```

Например, структура TNode

```
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 показанную на рисунке

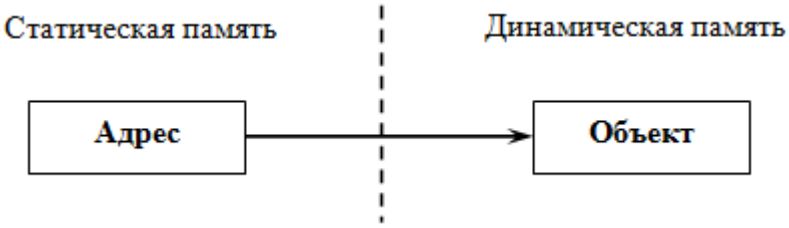


Данная структура состоит из 4 элементов. Ее первый элемент имеет поле Data, равное 73, и связан с помощью своего поля Next со вторым элементом, поле Data которого равно 28, и так далее до последнего, четвертого элемента, поле Data которого равно 85, а поле Next равно NULL, то есть нулевому адресу, что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

Доступ к данным в ДС

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:

```
указатель_на_структуру-> имя_элемента
```

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;
```

Имея возможность явного манипулирования с указателями, которые могут располагаться как вне структуры, так и "внутри" отдельных ее элементов, можно создавать в памяти различные структуры. Однако необходимо помнить, что работа с динамическими данными замедляет выполнение программы, поскольку доступ к величине происходит в два шага: сначала ищется указатель, затем по нему – величина.

Работа с памятью

В программах, в которых необходимо использовать динамические структуры данных, работа с памятью происходит стандартным образом. Выделение динамической памяти производится с помощью операции new, освобождение памяти – операции delete. Например, объявим динамическую структуру данных с именем Node и полями Name, Value и Next, выделим память под указатель на структуру, присвоим значения элементам структуры и освободим память.

```
struct Node
{
     char *Name;
     int Value;
     Node *Next;
};

Node *PNode; //объявляем указатель
PNode = new Node;      //выделяем память
```

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

delete PNode;         //освобождаем память
```

Задание

1) Разработайте динамическую структуру данных в соответствии со своим вариантом задания (см. список ниже). В качестве элемента ДС используйте шаблон класса TNode, параметром которого является тип хранимых в динамической структуре данных. Это позволит вам использовать разработанную ДС для различных типов (int, char, float и т.д.). Реализуйте операции вставки и извлечения элемента, вывода содержимого ДС на экран.
2) Проверьте работоспособность разработанной ДС. Для этого создайте пустую структуру, добавьте в нее несколько элементов, а затем удалите элементы из ДС. На каждом из указанных этапов содержимое выведите на экран.
3) Измерьте экспериментально функцию сложности f(n) для операций вставки и удаления элемента из ДС. Сравните полученный результат с теоретическим, полученным из анализа программного кода.
4) Исследуйте расположение элементов ДС в оперативной памяти. Для этого сделайте несколько операций добавления и удаления элементов в структуре, после чего выведите на экран адреса всех оставшихся элементов. Объясните полученный результат.

Варианты заданий

1) Односвязный список
2) Двусвязный список
3) Циклический (круговой) связный список
4) Стек
5) Очередь
6) Дек