1. **Списки**

Теперь, когда рассмотрены такие ключевые вопросы как структуры и файлы, можно перейти к обзору главной темы работы – динамические структуры данных. Однако, сперва рассмотрим отличия динамических структур данных, от статических:

*- Статические структуры* относятся к разряду непримитивных структур, которые, фактически, представляют собой структурированное множество примитивных, базовых, структур. Поскольку по определению такие структуры отличаются отсутствием изменчивости, память для них выделяется один раз и ее объем остается неизменным до уничтожения структуры. Простейшая структура данных такого типа - массив, где обращение к элементу происходит через его номер. В применении статических структур, есть как положительные стороны, так и отрицательные. К положительным можно отнести доступ за константное время к любому элементу, а к отрицательным - статичность, неизменность структуры.

*- 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 динамической структуры 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. Элемент динамической структуры состоит из двух полей - информационного поля или поля данных, в котором содержатся те данные, ради которых и создается структура, и поле связок, в котором содержатся один или несколько указателей, связывающий данный элемент с другими элементами структуры.

Достоинства у связного представления данных довольно многочисленны:

- лёгкость добавления и удаления элементов;

- размер ограничен только объёмом памяти компьютера и разрядностью указателей;

- при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;

Вместе с тем связное представление не лишено и недостатков, основные из которых:

- сложность определения адреса элемента по его индексу;

- на поля-указатели расходуется дополнительная память;

- элементы связной структуры могут быть расположены в памяти разреженно, что окажет негативный эффект на кэширование процессора;

- над связными структурами гораздо труднее производить параллельные векторные операции, такие как вычисление суммы.

Теперь, после рассмотрения разницы между статическими и динамическими структурами данных, можно вплотную разобраться с тем, что же такое список и какие виды динамических списков бывают.

*Списком* называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения. Список, отражающий отношения соседства между элементами, называется линейным. Длина списка равна числу элементов, содержащихся в списке, список нулевой длины называется пустым списком. Линейные связные списки являются простейшими динамическими структурами данных.

## Виды списков

* *Однонаправленный связный список* – ключевой особенностью данного вида списков является то, что ссылка в каждом узле указывает только на следующий узел в списке. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.
* *Двунаправленный связный список* - ссылки в каждом узле указывают на предыдущий и на последующий узел в списке. По двусвязному списку можно передвигаться в любом направлении — как к началу, так и к концу. В этом списке проще производить удаление и перестановку элементов, так как всегда известны адреса тех элементов списка, указатели которых направлены на изменяемый элемент.
* *XOR-связный список* — структура данных, похожая на обычный двусвязный список, однако в каждом элементе хранящая только один адрес — результат выполнения операции XOR над адресами предыдущего и следующего элементов списка. Для того, чтобы перемещаться по списку, необходимо взять два последовательных адреса и выполнить над ними операцию XOR, которая и даст реальный адрес следующего элемента.
* *Кольцевой связный список* – этот тип списков является разновидностью односвязных либо двусвязных списков. Последний элемент кольцевого списка содержит указатель на первый, а первый (в случае двусвязного списка) — на последний.
* *Список с пропусками* - вероятностная структура данных, основанная на нескольких параллельных отсортированных связных списках с эффективностью, сравнимой с двоичным деревом. В основе списка с пропусками лежит расширение отсортированного связного списка дополнительными связями, добавленными в случайных путях с геометрическим/негативным биномиальным распределением, таким образом, чтобы поиск по списку мог быстро пропускать части этого списка. Вставка, поиск и удаление выполняются за логарифмическое случайное время.
* *Развёрнутый связный* список - список, каждый физический элемент которого содержит несколько логических (обычно в виде массива, что позволяет ускорить доступ к отдельным элементам). Позволяет значительно уменьшить расход памяти и увеличить производительность по сравнению с обычным списком.

В практическом приложении к данной работе использовался однонаправленный связный тип списков, поэтому дальнейшее рассмотрение операций над списком, по причине избыточной информации в обратном случае, будет лишь над этим типом.

## 3.2 Создание списка

Для создания односвязного динамического списка в памяти, обязательными являются несколько ключевых моментов.

- Описание структурного типа, одним из полей которого будет указатель на структуру такого же типа;

- Создание указателя на начало списка – то есть, на структуру, описанную в предыдущем пункте;

В дальнейшем, при работе со списком, будет использоваться лишь этот указатель и описанный сначала структурный тип.

## 3.3 Просмотр списка

Для просмотра всего списка, то есть для каких-то действия требующих перемещения по всем его элементам, используется такой алгоритм:

- Создаётся временный указатель на элемент списка;

- Этому временному указателю присваивается значение указателя на начало списка. Это нужно для того, чтобы не потерять значение начального указателя;

- В цикле производится проход по всем элементам списка – при этом на каждой итерации цикла, временному указателю присваивается значение следующего элемента списка. Условие выхода из цикла - указатель на следующий элемент на равен значению null(Признак конца односвязного списка).

## 3.4 Изменение списка

Рассмотрение изменения элементов в односвязном списке можно разделить на несколько логических операций: добавление элементов в список, изменение существующего в списке элемента, вставка элемента в список, и удаление элемента из списка. Теперь можно подробнее рассмотреть каждый из них.

*Добавление элемента в список* - новая запись всегда добавляется в конец этого списка – то есть, если добавляется не первый элемент, требуется описанный выше алгоритм просмотра всего существующего списка. В результате этого алгоритма появляется временный указатель на последний элемент данного списка. В дальнейшем нужно, при помощи операции new выделить память под новый элемент списка и присвоить адрес выделенной памяти указателю на следующий элемент у последнего существующего в списке элемента. В дальнейшем, после присваивания полям нового элемента нужных значений, требуется присвоить значение null указателю на следующий (и пока не существующий) элемент.

Если же добавляется первый элемент списка, адрес выделенной операцией new памяти нужно присвоить указателю на начало списка.

*Изменение существующего в списке элемента* ***-***  используется проход по всему списку до нужного элемента (обычно определяется каким либо условием – какое-то определённое значение поля либо порядковым номером элемента в списке), затем просто изменяются поля данной записи, которые не содержат указатель на последующие элементы.

*Вставка элементов в список* ***-*** применяется немного изменённый алгоритм добавления. Сначала выполняется проход по списку до записи, которая будет предшествовать вставляемой записи. Затем, объявляется ещё один временный указатель и в него записывается адрес созданной функцией new записи. В указатель следующего элемента созданной записи записывается адрес следующего элемента предыдущей записи, и на место последнего записывается адрес вставляемой записи.

*Удаление элементов из списка -* применяется немного изменённый алгоритм прохода по всему списку – вместо одного временного указателя объявляется два. Второму временному указателю, на каждой итерации цикла будет присваиваться значение первого. По достижению нужной для удаления записи цикл прерывается, адресу указателя на следующий в списке элемент первого временного указателя присваивается адрес указателя на следующий в списке элемент второго временного указателя. Затем память по адресу первого временного указателя очищается функций delete. В случае если удаляется последний или первый элемент, применяется только один буферный указатель.