**7.1. Общие понятия**

Любая программа предназначена для обработки данных, от способа организации которых зависят алгоритмы работы, поэтому выбор структур данных должен предшествовать созданию алгоритмов. Ранее были рассмотрены стандартные способы организации данных, предоставляемые языком C++, — основные и составные типы. Наиболее часто в программах используются массивы, структуры и их сочетания.

Память под данные выделяется либо на этапе компиляции (в этом случае необходимый объем должен быть известен до начала выполнения программы, то есть задан в виде константы), либо во время выполнения программы с помощью операции new (необходимый объем должен быть известен до распределения памяти). В обоих случаях выделяется непрерывный участок памяти. Если до начала работы с данными невозможно определить, сколько памяти потребуется для их хранения, память выделяется по мере необходимости отдельными блоками, связанными друг с другом с помощью указателей. Такой способ организации данных называется *динамическими структурами данных,* поскольку их размер изменяется во время выполнения программы. Из динамических структур в программах чаще всего используются *линейные списки, стеки, очереди* и *бинарные деревья.* Они различаются способами связи отдельных элементов и допустимыми операциями. Динамическая структура может занимать несмежные участки оперативной памяти.

Динамические структуры широко применяют и для более эффективной работы с данными, размер которых известен, особенно для решения задач сортировки, поскольку упорядочивание динамических структур не требует перестановки элементов, а сводится к изменению указателей на эти элементы. Например, если в процессе выполнения программы требуется многократно упорядочивать большой массив данных, имеет смысл организовать его в виде линейного списка. При решении задач поиска элемента в тех случаях, когда важна скорость, данные лучше всего представить в виде бинарного дерева.

*Элемент* любой динамической структуры данных представляет собой структуру (в смысле struct), содержащую по крайней мере два поля: для хранения данных и для указателя. Полей данных и указателей может быть несколько. Поля данных могут быть любого типа: основного, составного или типа указатель. Описание простейшего элемента (компоненты, узла) выглядит следующим образом:

struct Node{

Data d; // тип данных Data должен быть определен ранее

Node \*р;

};

Рассмотрим реализацию основных операций с динамическими структурами данных.