**ЛЕКЦИЯ 8**

**АБСТРАКТЫНЕ КЛАССЫ И ВИРТУАЛЬНЫЕ ФУНКЦИИ**

**ВИРТУАЛЬНЫЕ ФУНКЦИИ**

1. Виртуальная функция – функция, которая в дочерних классах переопределяется;
2. Используется если требуется в дочернем классе определить свой вариант этой функции;
3. Классы, которые имеют такие функции, называются полиморфными. Полиморфизм – возможность функции изменять возвращаемое значение;
4. ВФ используют механизм позднего (отложенного, динамического) связывания. Любая не статическая функция может быть сделана виртуальной с помощью ключевого слова **virtual**;
5. ВФ могут быть как в базовом, так и в дочерних классах. Если ВФ не переопределять в дочерях базового класса, то она останется virtual;
6. Функция перестает быть ВФ после первого переопределения;
7. Виртуальные дизрукторы – гарантия его срабатывания.

**РАННЕ И ПОЗДНЕЕ СВЯЗЫВАНИЕ**

1. Раннее связывание: объект и функция связаны на этапе компиляции (обычные функции, перегруженные функции и операторы и т.д);
2. Позднее связывание: объект с функцией связываются на этапе исполнения программы. Связывания происходит через ВФ. Плюсы: гибкость за счет того, что разные объекты могут иметь собственный интерфейс или поддержаны общим интерфейсом.

**ПЕРЕОБПРЕДЕЛЕНИЕ ВИРТУАЛЬНОЙ ФУНКЦИИ**

1. Полиморфизм – использование наследования и виртуальных функций;
2. Чтобы переопределить функцию в дочерних классах, необходимо использовать ключевое слово override (оператор переопределения);
3. Статические функции виртуальными быть не могут, так как определяются всего один раз;

**ЧИСТО ВИРТУАЛЬНАЯ ФУНКЦИЯ**

1. **virtual <тип> <имя>() = 0;**
2. Чисто виртуальные функции не имеют определения;
3. ВФ будет чистой, если при объявлении присвоить 0;
4. ЧВФ должны быть определены во всех классах наследниках, иначе наследники становятся абстрактными классами.

**АБСТРАКТНЫЕ КЛАССЫ**

1. Содержит или наследует хотя бы одну ЧВФ без переопределения;
2. АК нельзя использовать когда идет явное приведение типов;
3. Нельзя использовать при описании типов параметра и типа значения, которое возвращает эта функция;
4. Если класс не переопределяет все ЧВФ родителя, то он тоже станет виртуальным.
5. Полиморфная функция – функция, способная обрабатывать данные разных типов (шаблоны).

**АТД КОНТЕЙНЕРЫ**

1. Абстракция – изменение подробностей задачи за счет упрощения;
2. Пример абстракции: использование любых библиотек;
3. Абстрактные типы данных – представление типа данных в виде списка операций, без способов реализации этих операций;
4. Параметризация – абстракция через параметры и структуры без привязки к типам данных;
5. Спецификация – описание того, как должна работать функция и что она должна реализовать;
6. Требование:
7. Должен работать только со своими типами данных (если мы делаем очередь, то необходимо сохранить её основные принципы);
8. Не должны зависеть от внешнего состояния приложения (должно работать корректно при любых манипуляциях);
9. Так же как и в классах, способы доступа производятся соответствующими методами;
10. Отличия АДТ и структур: структура – то, что находится внутри (реализация), АДТ – некоторая оболочка/описание для реализации;
11. Преимущества:
12. Инкапсуляция того, что пользователю трогать и видеть не надо;
13. Абстрагирования от ненужных деталей (декомпозиция);
14. Ограничение области использования данных (использования не для каждого класса);
15. Высокая информативность интерфейса за счет отсутцтвия мелкой детализации;

**КОНТЕЙНЕРЫ**

1. Контейнер – однотипный набор элементов (массив);
2. Сам контейнер – объект, имя – имя переменной;
3. Свойства:
4. Имеет время жизни, которое может не совпадать с временем жизни элементов контейнера;
5. Элементами контейнера могут быть другие контейнеры и объекты;
6. Контейнеры могут быть фиксированного и переменного размера;

**ВИДЫ КОНТЕЙНЕРОВ**

1. Last in First out – последний пришел первый вышел;
2. First in Last out – первым пришел последним вышел;

**ОПЕРАЦИИ КОНТЕЙНЕРОВ**

1. Замена значений через операцию доступа;
2. Операция добавления и удаления элементов или групп элементов;
3. Поиск элементов или групп элементов;
4. Операция объединения контейнеров;
5. Специальные операции, которые зависят от вида контейнеров (детерминант для матрицы);

**ИТЕРАТОРЫ**

1. Является абстрактным типов данных для доступа к элементам контейнера и перехода от одного к другому;
2. Назначение: обращение к любому элементу без привязки к структуре контейнера;
3. Хранение элементов любым способом;
4. Дает доступ к элементу, содержит функции для перехода к другим элементам;
5. У итератора есть доступ к первому элементу контейнера. В него заложена проверка на конечность контейнера.

**ШАБЛОНЫ ФУНКЦИЙ И КЛАССОВ**

1. Шаблоны классов и типов – создание функций или классов, которые работают с разными типами данных;
2. Создают семейства функций и классов;
3. Создание функций и классов автоматизируется (единый алгоритм) для данных разного типа;
4. Параметр шаблона- тип данных;
5. Шаблоны функция и класса называются параметризованными;
6. Реализация шаблонов функций и классов называется обобщенным программированием;

**ШАБЛОНЫ ФУНКЦИЙ**

1. Это описание функции без типа данных. Тип передает как параметр;
2. Компилятор генерирует функцию для каждого типа.

**ОСНОВНЫЕ СВОЙСТВА ПАРАМЕТРОВ ШАБЛОНОВ**

1. В списке параметров должно быть ключевое **слово typename или class**;
2. Переменные типа Type, T и так далее имеют все прочие свойства переменных;
3. Один шаблонный тип данных может в одной перегрузке “превратиться” только в один тип данных;
4. Имена параметров в шаблоне не обязаны совпадать с именами в самом шаблоне.

**ШАБЛОНЫ КЛАССОВ**

1. Используется для создания родственного класса;
2. При описании родового класса можно создать семейство родственных;
3. АТД может использоваться в качестве параметра шаблона;
4. Параметризованное имя семейства классов может быть одним для всего семейства;
5. Инстанцирование шаблона – генерация класса по шаблону.

**СВОЙСТВА ШАБЛОНОВ**

1. Параметризация распространяется на все методы и поля класса;
2. Дружественные функции шаблонного класса не являются параметризованными;
3. Если дружественная функция принимает в себе дружественный класс, то каждая instance класса имеют свою дружественную функцию;
4. В параметризованном классе нельзя объявить дружественный параметризованный класс;
5. Шаблоны могут быть базовыми классами (наследникам не надо быть шаблонами);
6. Шаблоны функций – членом нельзя описывать как виртуальные;
7. Нельзя делать шаблонный функции во внутреннем классе;
8. Для объявления типа данных модно использовать T::<тип\_данных> переменная.