Событийно-ориентированное программирование (event-driven programming) — это парадигма программирования, в которой выполнение программы определяется событиями — действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета).

Событийно-ориентированное программирование можно также определить как способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события.

Событийно-ориентированное программирование, как правило, применяется в трех случаях:

1. при построении пользовательских интерфейсов (в том числе графических);
2. при создании серверных приложений в случае, если по тем или иным причинам нежелательно порождение обслуживающих процессов;
3. при программировании игр, в которых осуществляется управление множеством объектов.

**Событийно-ориентированное программирование.**

С активным распространением системы *Windows*и появлением визуальных *RAD-*сред широкую популярность приобрел событийный подход к созданию программ – *событийно-ориентированное программирование.*

Идеология системы *Windows*основана на событиях. Щелкнул человек на кнопке, выбрал пункт меню, нажал на клавишу или кнопку мыши — в *Windows*генериру­ется подходящее *сообщение,*которое отсылается окну соответствующей программы.

Структура программы, созданной с помощью событийного программирования, следующая. Главная часть представляет собой один бесконечный цикл, который опрашивает *Windows,*следя за тем, не появилось ли новое сообщение. При его обна­ружении вызывается подпрограмма, ответственная за *обработку*соответствующего события (обрабатываются не все события, их сотни, а только нужные), и подобный цикл опроса продолжается, пока не будет получено сообщение «Завершить работу».

События могут быть *пользовательскими,*возникшими в результате действий поль­зователя, *системными,*возникающими в операционной системе (например, сообще­ния от таймера), и*программными,*генерируемыми самой программой (например, обнаружена ошибка и ее надо обработать).

Событийное программирование является развитием идей нисходящего проекти­рования, когда постепенно определяются и детализируются реакции программы на различные события.

**Событийно-ориентированное программирование**– это способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется **главный цикл приложения**, тело которого состоит из двух частей: **выборки события** и **обработки события**.

Событийно-ориентированное программирование, как правило, применяется в трех случаях:

при построении пользовательских интерфейсов (в том числе графических);

при создании серверных приложений в случае, если по тем или иным причинам нежелательно порождение обслуживающих процессов;

при программировании игр, в которых осуществляется управление множеством объектов.

Примеры реализаций:

Веб-серверы:

nginx

lighttpd

Прокси-серверы:

Squid

Применение в интерфейсах. СОП может быть как расширением структурного, так и объектно-ориентированного программирования (в дальнейшем ООП). Эпостасиями СОП являются cобытия и их приоритеты. События — специльные процедуры (для структурного програмирования) или методы (для ООП). События можно разделить на обычные, выполнение которых начинается по их прямому вызову (пользователя или программы) и события-демоны, находящихся в фоновом режиме и выпняющихся при совершении события, например, выполнения условий. При наступления ситуации, когда событие должно начать работу оно уступает событиям, имеющий больший приоритет. Приоритет может быть выражден от динамического до статического, крайним случаем которого является порядок следования событий.

Языки программирования. Разные языки программирования поддерживают СОП в разной степени. Наиболее полную поддержку событий обладают следующие языки (неполный список):

Perl (события и демоны DAEMON, и их приоритеты PRIO),

Java,

Delphi,

C# (события event).

Классы событий могут наследоваться (для исключений необходимо определить класс, как класс производный от системного: у С# системным является Exception, у С++ ).

Остальные языки, в большей их части, поддерживают события как обработку исключительных ситуаций.