ПРИМЕНЕНИЕ СОБЫТИЙНО-ОРИЕНТИРОВАННОГО ПОДХОДА К ПОСТРОЕНИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРИ ДЛИТЕЛЬНОМ ВРЕМЕНИ ОБРАБОТКИ СОБЫТИЯ

Пономарева А.В., Шамин А.А.

Томский политехнический университет, Институт кибернетики
apon1993@gmail.com

Введение

Событийно-ориентированное программирование – парадигма программирования, в которой выполнение программы определяется событиями – действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы. Это способ написания кода программы, в котором можно выделить главный цикл, тело которого состоит из выбора события и его обработки [1].

Некоторые примеры областей, в которых оправдано применение событийного подхода к созданию программ:

- построение пользовательских интерфейсов (в том числе графических интерфейсов);
- параллельные вычисления;
- создание серверных приложений;
- автоматические и автоматизированные системы управления;
- создание игр, в которых необходимо управлять множеством объектов;
- и др.

Структура событийно-ориентированной программы

Событийно-ориентированные программы имеют типовую структуру (рис. 1), состоящую из трёх частей: секции инициализации программы, цикла выборки-обработки событий и завершающей секции программы.

Входными данными цикла обработки событий служат: очередь событий (S1, S2, …, Sn); таблица обработчиков событий (P1, P2, …, Pm).

Очередь событий имеет переменный размер, зависящий от скорости возникновения событий. Таблица обработчиков событий содержит указатели на процедуры обработки событий.

После извлечения события из очереди следует процесс поиска обработчика данного события. Если для полученного события не найден соответствующий обработчик, то оно игнорируется. Если обработчик найден, то событие обрабатывается.

Если очередь событий пуста, то программа переходит в режим ожидания и находится в этом режиме до возникновения события в очереди событий. В режиме ожидания программа, как правило, не потребляет ресурсов процессорного времени. Таким образом, если микропроцессорная система выполняет несколько задач, запрограммированных согласно событийно-ориентированной парадигме, то процессорное время будет тратиться только на те из них, которые в данный момент обрабатывают события. Это неоспоримое достоинство событийно-ориентированного подхода к построению программного обеспечения, позволяющее рационально использовать ресурс процессорного времени.

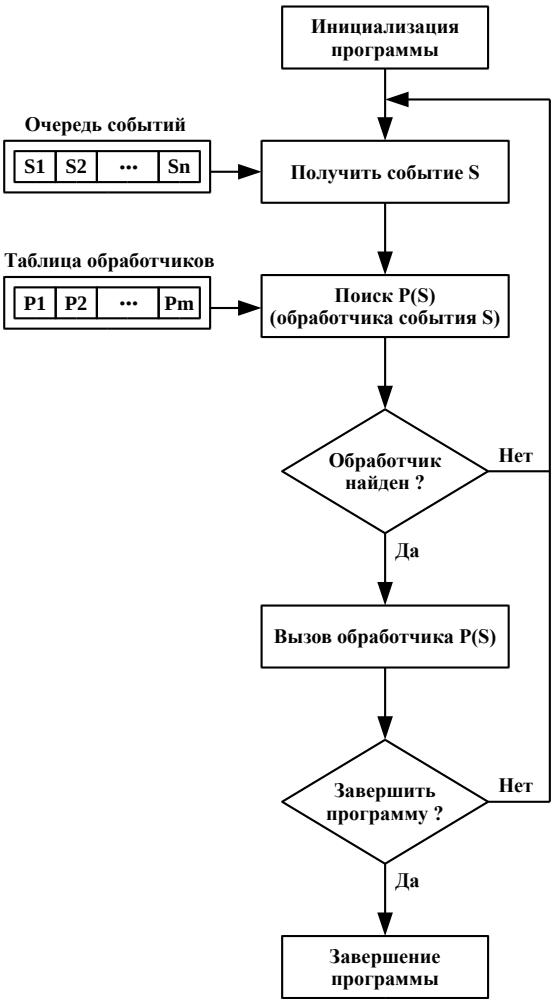

Рис. 1. Типовая структура событийно-ориентированной программы

Модификация структуры событийно-ориентированной программы

К недостаткам рассматриваемого подхода относится невозможность обработки событий, требующей длительного времени. Это обусловлено тем, что пока обрабатывается одно событие, программа не реагирует на другие возникшие

события. В результате увеличивается время отклика программы на возникшее событие и, как следствие, возникает опасность переполнения очереди событий. Подобная проблема возникает, например, при приеме большого файла или запуске внешнего процесса, ожидание завершения которого занимает достаточно большое время.

Часто устранить этот недостаток возможно, если обработчик события, требующего длительной обработки, построить по принципу конечного автомата (рис. 2) [2].

Предположим, что нам требуется по приему некоторого события **S**, запустить процесс **P**, затем дождаться завершения процесса **P**. В таком случае состояния обработчика будут следующими: готовность обработчика (***ready***), ожидание завершения процесса **P** (***wait***), завершение обработки процесса **P** (***done***).

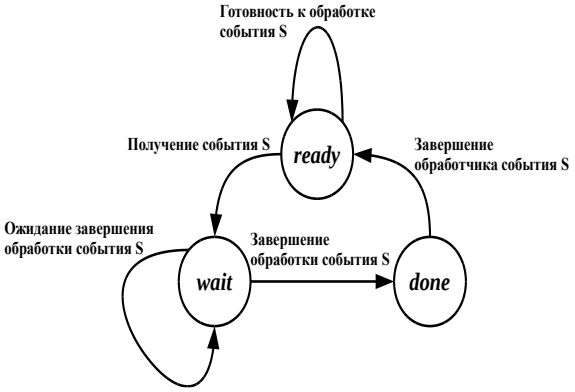

Рис. 2. Граф состояний конечного автомата обработки событий

Введем событие ***idle***, которое периодически генерируется по таймеру с определённым периодом. При возникновении события **S** обработчик данного события **Ps** переходит из состояния ***ready*** в состояние ***wait***. Обработчик события ***idle*** вызывает методы проверки окончания состояния ожидания для всех обработчиков **P1 … Pm**, находящихся в состоянии ***wait.***

Если процесс **P** обработки события **S** в обработчике **Ps** завершился, то обработчик **Ps** переходит из состояния ***wait*** в состояние ***done***, «обработка события **S** завершена». Затем обработчик ***Ps*** вновь переходит из состояния ***done*** в состояние ***ready.***

Модифицированная структура событийно-ориентированной программы представлена на рис. 3. От структуры программы, представленной на рис. 1, данная структура отличается наличием источника событий ***idle*** (таймером) и наличием обработчика событий ***idle*** (**Pidle**).

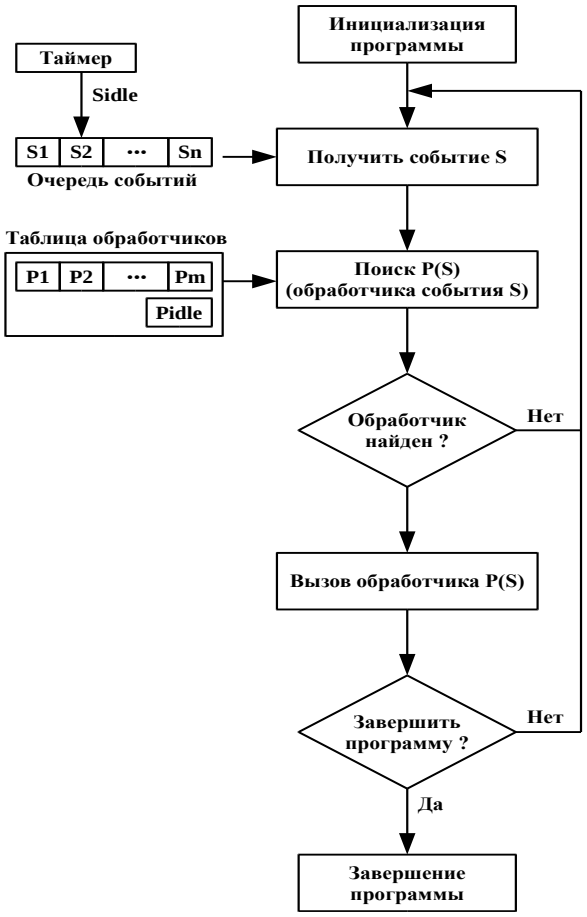

Рис. 3. Модифицированная структура событийно-ориентированной программы

Заключение

Предложенный подход к построению обработчиков событий по принципу конечного автомата апробирован в нескольких программах на языке C++. В результате апробации выявлено, что данный подход является рациональным при построении программного обеспечения для обеспечения пользовательского интерфейса на встраиваемых устройствах.

Список использованных источников

1. Событийно-ориентированное программирование. [Электронный ресурс]. – Режим доступа: http://bourabai.ru/alg/sop.htm, свободный (дата обращения: 19.10.2015).
2. Туккель Н.И., Шалыто А.А. Программирование с явным выделением состояний // Мир ПК. - 2001. - №9 - С. 132-138