Н.Н. Красильников, krasilnikov@rain.ifmo.ru

Реализация автоматов на основе указателей на методы

Изучая реализацию библиотеки Boost, я наткнулся на достаточно занятную реализацию автомата. Ею я и хочу поделиться.

Найденная реализация основана на указателях на методы класса. Состоянием автомата в ней является не число, а метод, который будет вызван при обработке события. Текущее состояние хранится в указателе на соответствующий метод.

Данный подход позволяет заменить поиск текущего состояния (например, при помощи оператора switch) на простой вызов метода, на который ссылается указатель текущего состояния.

Далее приводится пример подобной реализации (Приложение 1).

Приложение 1. Пример

```
// Автомат

class Automata {
    // Типы состояния, входных переменных и выходных воздействий
    typedef void State(int e);
    typedef void Output();
    typedef bool Input();
    typedef void (Automata::*Inner)(int);

public:
    // Текущее состояние автомата
    Inner y;

    // Состояния
    State s0;
    State s1;

    // Входные переменные
    Input x0;

    // Выходные воздействия
    Output z0;
    Output z1;

    // Конструктор
    Automata();
    // Обработчик событий
    void Main(int e);
};
```

```
// Конструктор
Automata::Automata() {
    // Задаем начальное состояние
    y = &Automata::s0;
}

// Обработчик событий
void Automata::Main(int e) {
    // Вызов обработчика текущего состояния
    (this->*y)(e);
}

// Состояния

void Automata::s0(int e) {
    if (e == 0 && x0()) {
        z0();
        // Переход в состояние s1
        y = &Automata::s1;
    } else if (e == 1) {
        z1();
    }
}

void Automata::s1(int e) {
    if (e == 0 && x0()) {
        z0();
    } else if (e == 1) {
        z1();
        // Переход в состояние s0
        y = &Automata::s0;
    }
}

// Входные переменные

bool Automata::x0() {
    return true;
}

// Выходные воздействия

void Automata::z0() {
}

void Automata::z1() {
}
```

```
// Пример использования

int main() {
    Automata a;
    a.Main(0);
    a.Main(1);
    return 0;
}
```