CMPE-013/L

Introduction to “C” Programming

Bryant Wenborg Mairs

Spring 2014
Switch statements
**switch Statement**

Real-world example

```c
bool IsHex(char character) {
    switch (character) {
    case 'a' ... 'f':
    case 'A' ... 'F':
    case '0' ... '9':
        return true;
    default:
        return false;
    }
}
```
switch Statement

Switch versus if/else

• Subset of functionality of if/else
• Works in cases with one value to test
• Switches can be more compact
  – No need to retype variable being tested
  – Range syntax (for GCC-compatible compilers)
  – Easier to read when used properly
• Good when combined with enums
**switch Statement**

With enums

```c
typedef enum {
    PARAMEVENTNONE,
    PARAMEVENTREQUESTLISTRECEIVED,
    PARAMEVENTREQUESTREADRECEIVED,
    PARAMEVENTSETRECEIVED
} ParamEvent;
```

```c
ParamEvent x = PARAMEVENTNONE;
switch (x) {
    case PARAMEVENTNONE:
        puts("PARAMEVENTNONE found.");
        break;
    case PARAMEVENTREQUESTLISTRECEIVED:
        puts("PARAMEVENTREQUESTLISTRECEIVED found.");
        break;
}
```
switch Statement

With enums

Example

```c
ParamEvent x;
switch (x) {
    case PARAM_EVENT_NONE:
        puts("PARAM_EVENT_NONE found.");
        break;
    case PARAM_EVENT_REQUEST_LIST_RECEIVED:
        puts("PARAM_EVENT_REQUEST_LIST_RECEIVED found.");
        break;
}
```

Errors

```
test.c:141:1: warning: enumeration value 'PARAM_EVENT_REQUEST_READ_RECEIVED' not handled in switch
test.c:141:1: warning: enumeration value 'PARAM_EVENT_SET_RECEIVED' not handled in switch
```
switch Statement

With enums

Example

```c
ParamEvent x;
switch (x) {
    case PARAM_EVENT_NONE:
        puts("PARAM_EVENT_NONE found.");
        break;
    case PARAM_EVENT_REQUEST_LIST_RECEIVED:
        puts("PARAM_EVENT_REQUEST_LIST_RECEIVED found.");
        break;
    default:
        break;
}
```
switch Statement

Local variables

Example

```c
int x;
switch (x) {
    case 3:
        int i;
        for (i = 0; i < 3; ++i) {
            puts("x");
        }
        break;
    ...
}
```

Errors

**error:** a label can only be part of a statement and a declaration is not a statement
**switch Statement**

Local variables

**Example**

```c
int x;
switch (x) {
    case 3:
        {  
            int i;
            for (i = 0; i < 3; ++i) {  
                puts("x");  
            }
        }
    break;
    ...  
}
```

```c
SET_HOME:

 mavlink_msg_home_t i;
 mavlink_decode(&m, &i);
```

State machines
State machines

- Known as Finite State Machines (FSM)
- Mathematical model of computation where system has a single state
- Triggering conditions can change that state
- FSMs are defined completely by both their states and the transitions between them
State machines

State

- The system only exists in one state at a time
- State persists through time
- Certain conditions can change the state to another state
  - These are specific to the current state
State machines

Transitions

• Events trigger transitions between states
• A combination of events can be used
• Transitions are all mutually exclusive
• At any given time there must be a valid transition for a state
  – If no transition is explicitly stated, an implied loopback transition exists
State machines

Benefits

• Provides a formal way to reason about a system
  – Allows for testing before writing any code
• Can be easily visualized
• Are language independent
• States are only dependent on current state and current inputs
State machines

When to use

• Can be used whenever there are a finite set of states for the system
  – Car transmission
  – Stoplight
  – Vending machine
  – Toaster oven
  – Video games
State machines

Use in the SeaSlug

- Transmission protocol
  - Mission management
  - Parameter management
- Operating state
  - Handling errors/system faults
- Calibration
  - Rudder
  - Radio controller
State machines

Diagrams

STATE_1

initial transition

STATE_2

action 3

condition 1

action 1

action 2

condition 2

Timer Event & Button Event (up)

action 4

action 5
State machines
Coding

Example

typedef enum { STATE_1, STATE_2 } SystemState;
static SystemState state;
{
    switch (state) {
    case STATE_1:
        if (condition1) {
            Action1();
            state = STATE_2;
        }
        break;
    case STATE_2:
        if (condition2) {
            Action2();
            state = STATE_1;
        }
    }
}
typedef enum { STATE_1, STATE_2 } SystemState;
static SystemState state;
int main (void) {
    // Initialize system

    // Event loop
    while (1) {
        // State machine
        switch (state) {
            ...
        }
    }
}
State machines

Bounce lab example
State machines

Bounce lab example

• Live coding example!
State machines
Bounce lab example

• State machines rely on conditions to trigger state machines
  – Fits in nicely with event-driven programming
• Use an enum datatype for your states and a switch-statement to check them
  – Provides some compile-time checks
State machines

Bounce lab example

• Bounce left: the timer expires and SW1 on
• Bounce right: the timer expires and SW2 on
State machines
Bounce lab example

• Live coding example!