

System-Level Programming

8 Control Structures

J. Kleinöder, D. Lohmann, V. Sieh, P. Wägemann

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Summer Term 2024

<http://sys.cs.fau.de/lehre/ss24>



goto Instruction

```
...  
goto Label
```

```
Label:  
...
```

- `goto` statement rarely used in clean code
- Edgar Dijkstra: „Go To Statement Considered Harmful”

```
Label:  
...
```

```
goto Label  
...
```

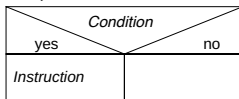
- `goto` leads to hard-to-read code
- Label must not be the function’s last statement

- `goto` and `if (...) goto` statements are the only control structures that are hardware can directly execute.
- This aspect is essential for understanding interrupts!



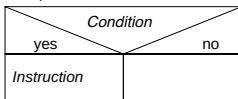
- `if` statement (conditional statement)

```
if (condition)  
    instruction;
```



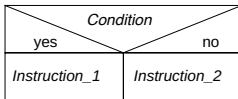
- `if` statement (conditional statement)

```
if (condition)
    instruction;
```



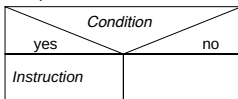
- `if-else` statement (two branches)

```
if (condition)
    instruction1;
else
    instruction2;
```



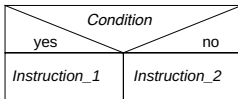
- `if` statement (conditional statement)

```
if (condition)
    instruction;
```



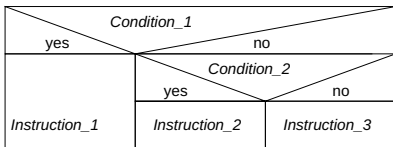
- `if-else` statement (two branches)

```
if (condition)
    instruction1;
else
    instruction2;
```



- `if-else-if` cascade (multiple branches)

```
if (condition1)
    instruction1;
else if (condition2)
    instruction2;
else
    instruction3;
```



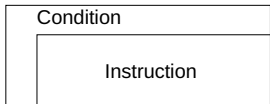
- `switch` statement (case differentiation)
 - alternative to `if` cascade when testing for integer values

integer expression = ?				
Value1	Value2			else
Inst. 1	Inst. 2		Inst. n	Inst. x

```
switch (expression) {  
  case value1:  
    instruction1;  
    break;  
  case value2:  
    instruction2;  
    break;  
  ...  
  case valuen:  
    instructionn;  
    break;  
  default:  
    instructionx;  
}
```



- Pre-condition loop
 - `while`-loop
 - executed zero or more times



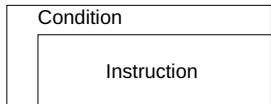
```
while(condition)
    instruction;
```

```
while (
    sb_button_getState(BUTTON0)
    == RELEASED
) {
    ... // do unless button press.
}
```



■ Pre-condition loop

- `while`-loop
- executed zero or more times

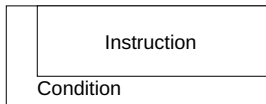


```
while(condition)
    instruction;
```

```
while (
    sb_button_getState(BUTTON0)
    == RELEASED
) {
    ... // do unless button press.
}
```

■ Post-condition loops

- `do-while` loops
- executed once or more



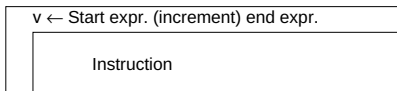
```
do
    instruction;
while(condition);
```

```
do {
    ... // do at least once
} while (
    sb_button_getState(BUTTON0)
    == RELEASED
);
```



- **for** loop (loop with explicit counter)

```
for (starting_expression;  
    terminating_expression;  
    incrementing_expression)  
    instruction;
```



- Example (usually: n executions with counter variable)

```
uint8_t sum = 0; // calc sum 1+...+10  
for (uint8_t n = 1; n < 11; n++) {  
    sum += n;  
}  
sb_7seg_showNumber( sum );
```



- Remarks

- Declaring a variable (n) in the *starting_expression* is only possible from C99 onwards.
- The loop is repeated as long as *terminating_expression* $\neq 0$ (*true*)
↪ the **for** loop is a more explicit **while** loop



- The current iteration of the loop can be terminated with the `continue` instruction.

↪ The loop continues with the next iteration

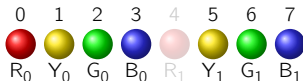
```
for (uint8_t led = 0; led < 8; led++) {  
    if (led == RED1) {  
        continue;           // skip RED1  
    }  
    sb_led_on(led);  
}
```



- The current iteration of the loop can be terminated with the `continue` instruction.

↪ The loop continues with the next iteration

```
for (uint8_t led = 0; led < 8; led++) {  
    if (led == RED1) {  
        continue;           // skip RED1  
    }  
    sb_led_on(led);  
}
```



- The execution of the whole (innermost) loop is terminate with the `break` instruction.

↪ The program resumes execution *after* the loop

```
for (uint8_t led = 0; led < 8; led++) {  
    if (led == RED1) {  
        break;             // break at RED1  
    }  
    sb_led_on(led);  
}
```



Loops & goto Instructions

- All loop types have semantically-equivalent sequences with `goto` statements
- Example:

```
for (uint8_t led = 0; led < 8; led++) {  
    if (led == RED1) {  
        continue; /* skip RED1 */  
    }  
    sb_led_on(led);  
}
```

```
uint8_t led = 0;  
goto test;  
loop:  
    if (led == RED1)  
        goto next;  
    sb_led_on(led);  
next:  
    led++;  
test:  
    if (led < 8)  
        goto loop;  
end:
```

