Programming with Structs

Structs

A real example on how we use `struct` in controlling a train.

```
#define MAXMSG  2


struct Message                // buffer for command
{
  unsigned char data[7];    // number of possible
  unsigned char len;        // number of positions in use
};


struct Message msg[MAXMSG]=
{
  { { 0xFF, 0,0xFF, 0, 0, 0, 0}, 3},    // idle msg
  { {    0, 0,    0, 0, 0, 0, 0}, 3},    // loco Msg
};
```

In this struct we have a collection of variables.

- the first line is a idle message to the train,
- the next line is right now empty,

In the program we are changing the first 3 zeros in line two, the first zero will contain the loco address, ex 36, the next will be a command ex 129 (light on, or 128 light off) The third zero will have to be a value where the address are “exclusive or” by the command this is used in the train as a error code. Every time we want to send a command to the train we will have to change these three values.

We are addressing the variables like this

```
 msg[1].data[0] = locoAddr1;
msg[1].data[1] = data1;
msg[1].data[2] = xdata1;
```

You are able to use pointers here as well.

Assignment 13

Make a C program which can print out the content of a `struct` as mentioned above.

This is going to be used for test purposes.

Assignment 14

Make a C program which can set the values for train number 7 and turn off the light.

Test the result by using the result of Assignment 13.

Assignment 15

Make a C program with an array containing `structs`.

I.e. an array with orders to trains. Fill it with train numbers 1, . ., (size_of_array) and orders to turn the light of.

Make the reference to the array circular. Simulate, that you send the orders with printout and send them one by one. Remember use pointer(s) !

For every traversal of the array the lights must be turn on/off (the opposite of what it was).

Assignment 16

Make a procedure that can exchange one of the orders in the array with a new f.ex. the idle order.

Remember to check the result with printout.

Code organization

Assignment 17

Organize the code developed in Assignments 13 – 16 in a *.c and *.h file for future use as a library.

Test the use of it with a new main() program.

Memory Allocation

Beside forwarding orders to train, we are going to do it to signals and switches too. And we are going to do it depending upon the positions of all the trains. A typical state decision problem.

Assignment 18

In a new program allocate memory by using Malloc() to the array from Assignment 15.

Make a new `struct` that contains :

- train number
- train position now
- train position next

where position is a number in the interval from 1 to 16. Initialize the positions with random values.

Allocate memory by using `Malloc()` to this `struct` too.

Assignment 19

Construct a test procedure for train positions, where you use pointers.

Assignment 20

Simulate over a number of time steps using pointers the position change of the trains.

> If two (or more) trains has the same “next position”,
>
> they do not move, but get a new “next position”

or

> if a train is the only one wanting a “next position”,
>
> it changes to that position