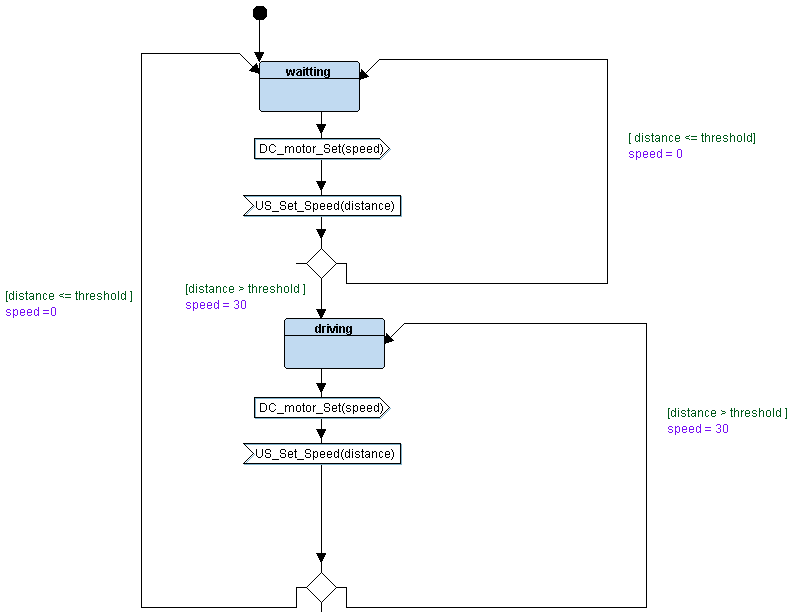
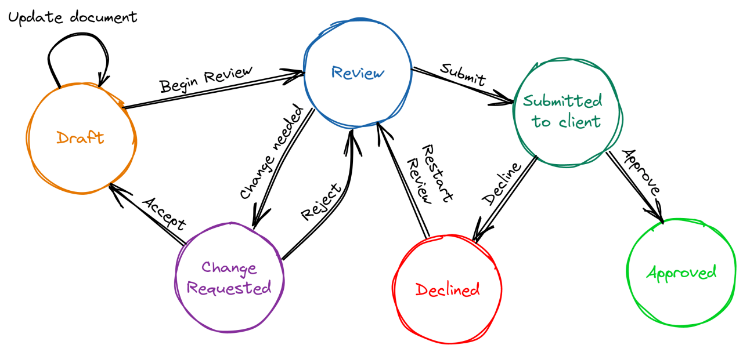
**State Machines in C**

a state machine is a design pattern used to manage and transition between different states of a system based on inputs or events. This pattern is particularly useful for implementing complex logic where the system needs to respond differently depending on its current state.

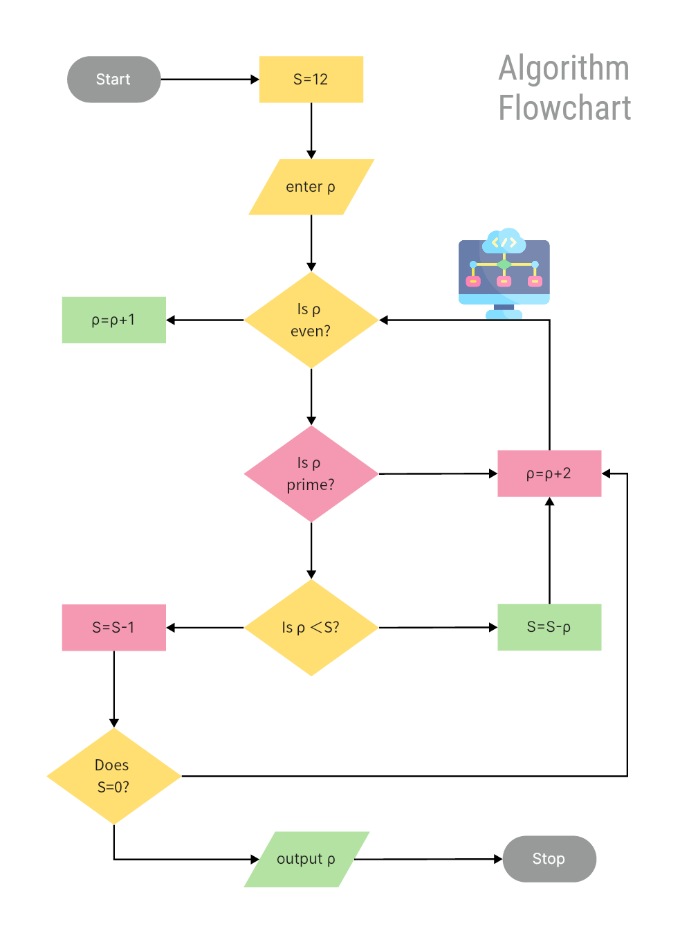
****

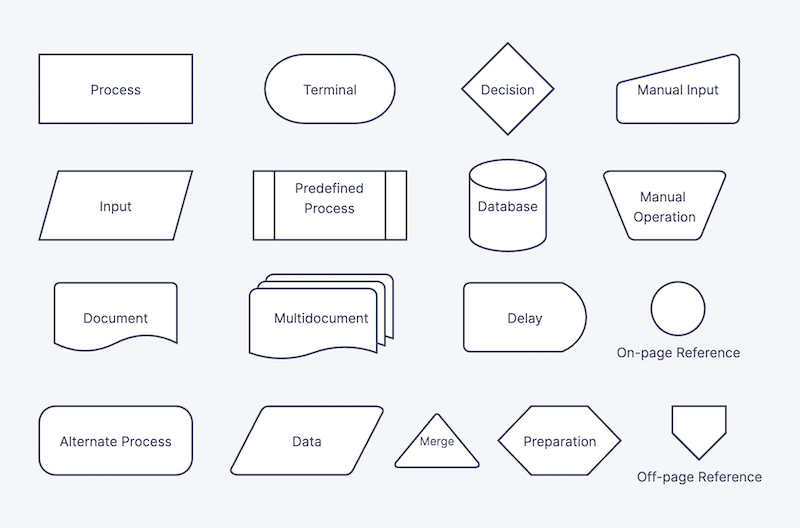


**Key Concepts**

1. **States**: Represent the different conditions or modes the system can be in. Each state may have specific behaviors or actions associated with it.
2. **Transitions**: Define how the system moves from one state to another based on certain conditions or inputs.
3. **Events**: Trigger transitions between states. Events are typically inputs or conditions that cause the system to change its state.

**Flowchart symbols**

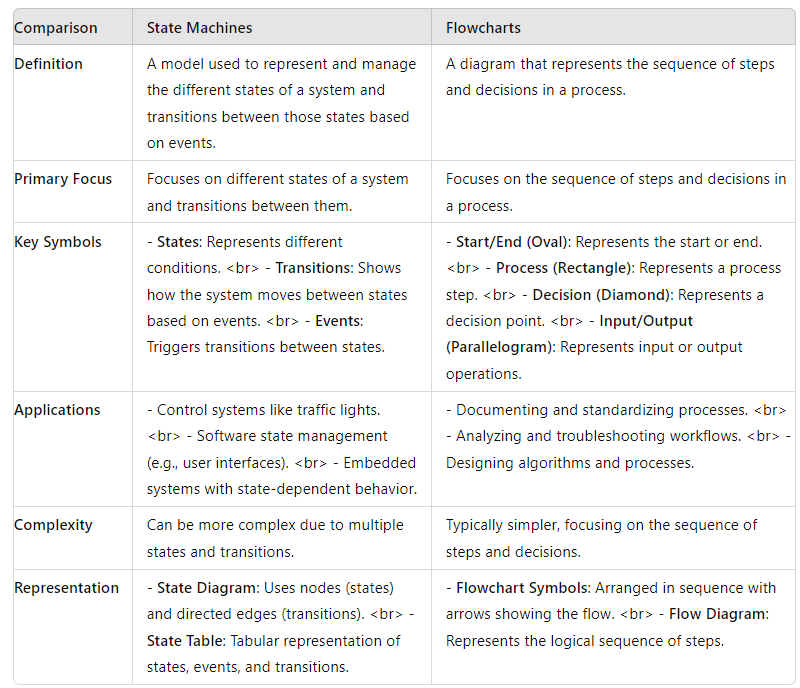
A flowchart is a diagrammatic representation used to visualize the sequence of steps and decisions in a process or system. It helps in understanding, designing, and analyzing processes by breaking them down into individual components and illustrating the flow of control between them.



### Key Features of Flowcharts

1. **Visual Representation**: Flowcharts use symbols and shapes to represent different types of actions, decisions, and processes, making complex workflows easier to understand.
2. **Step-by-Step Process**: They illustrate each step in a process, from the start to the end, showing how inputs are transformed into outputs.
3. **Decision Points**: Flowcharts highlight decision points where different paths can be taken based on certain conditions.
4. **Flow Direction**: Arrows or lines indicate the direction of flow between different steps or actions.

**State Machine vs Flowchart**

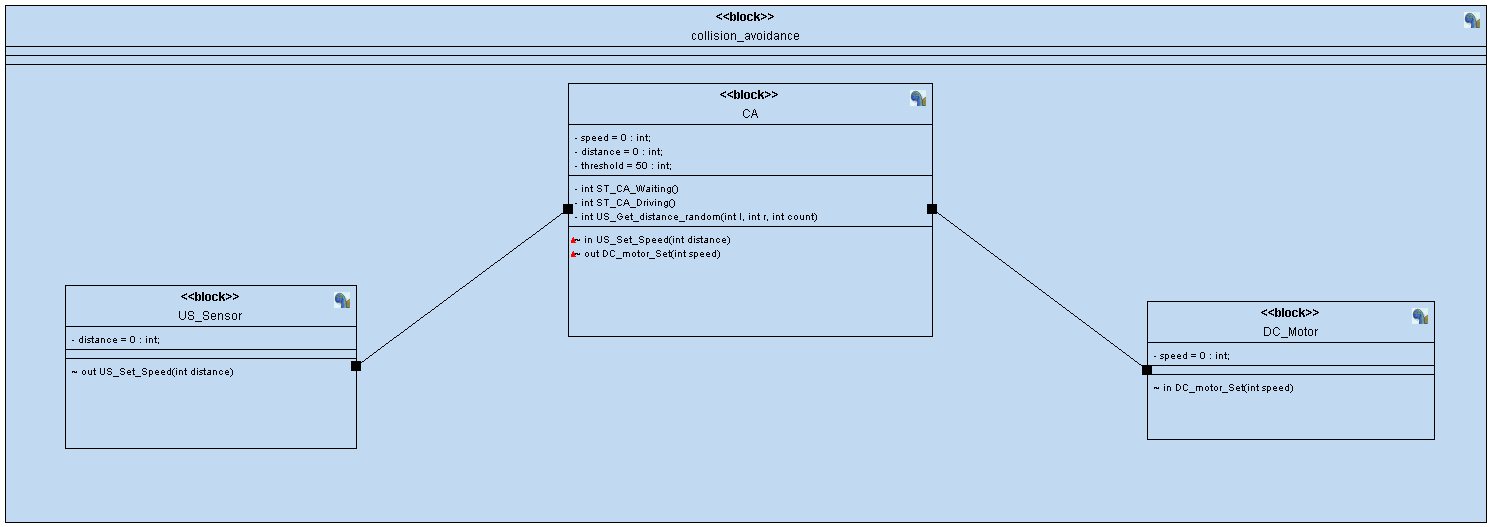


**types of state machines**

* **Finite State Machine (FSM)**: Basic state machine with a finite number of states.
* **Moore Machine**: Output depends only on the current state.
* **Mealy Machine**: Output depends on both the current state and input.
* **Hierarchical State Machine**: States are nested within other states.
* **Extended State Machine**: Includes additional variables affecting transitions.
* **Probabilistic State Machine**: Uses probabilities for state transitions.
* **Timed State Machine**: Transitions are based on time constraints.

**Implement Simple state machine in C**

**using multiple Modules**

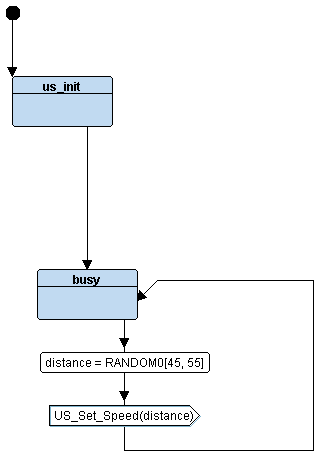
****

US\_Sensor

/\*

\* CA.c

\*

 \* Created on: Aug 4, 2024

\* Author: Abdallah Ghazy

\*/

**#include** "US.h"

**int** US\_distance = 0;

**int** **US\_Get\_distance\_random**(**int** l, **int** r, **int** count) {

**return** (**rand**() % (r - l + 1)) + l;

}

**void** (\*US\_state)();

**US\_init**(){

**printf**("\nUS\_init......");

}

STATE\_define(US\_busy) {

US\_state\_id = *US\_busy*;

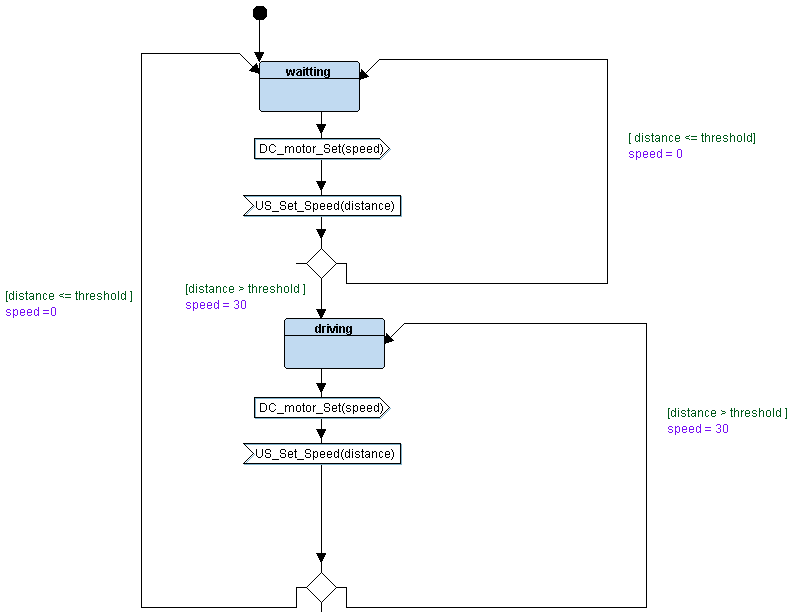
US\_distance = **US\_Get\_distance\_random**(45, 55, 1);

**printf**("\n CA\_Waiting State: distance = %d Speed = %d \n", US\_distance);

**US\_set\_distance**(US\_distance);

US\_state = STATE(US\_busy);

}

Collision avoiding

/\*

\* CA.c

\*

\* Created on: Aug 4, 2024

\* Author: Abdallah Ghazy

\*/

**#include** "CA.h"

**int** CA\_speed = 0;

**int** CA\_distance = 0;

**int** CA\_threshold = 50;

**void** (\*CA\_state)();

**void** **US\_set\_distance**(**int** d){

CA\_distance = d;

CA\_state = (CA\_distance <= CA\_threshold) ? STATE(CA\_waiting) : STATE(CA\_driving);

**printf**("US-------------- Distance = %d -------------------------- \n",CA\_distance);

}

STATE\_define(CA\_waiting) {

CA\_state\_id = *CA\_waiting*;

**printf**("\n CA\_Waiting State: distance = %d Speed = %d \n", CA\_distance, CA\_speed);

CA\_speed = 0;

**DC\_motor**(CA\_speed);

}

STATE\_define(CA\_driving) {

CA\_state\_id = *CA\_driving*;

CA\_speed = 30;

**printf**("\n CA\_Driving State: distance = %d Speed = %d \n", CA\_distance, CA\_speed);

**DC\_motor**(CA\_speed);

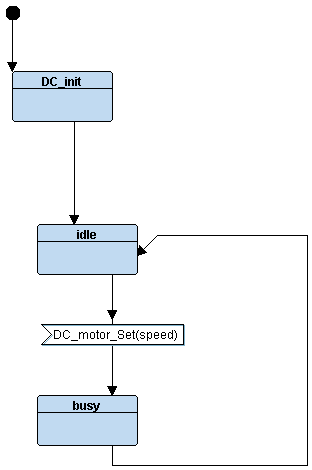
}

**DC\_Motor**

/\*

\* CA.c

\*

**** \* Created on: Aug 4, 2024

\* Author: Abdallah Ghazy

\*/

**#include** "DC.h"

**#include** "state.h"

**int** DC\_speed = 0;

**void** (\*DC\_state)();

**void** **DC\_init**() {

**printf**("DC\_init.............");

}

**void** **DC\_motor**(**int** s) {

DC\_speed = s;

DC\_state = STATE(DC\_busy);

**printf**("US-------------- DC\_speed = %d -------------------------- \n",

DC\_speed);

}

STATE\_define(DC\_idle) {

DC\_state\_id = *DC\_idle*;

**printf**("\n DC\_idle State: Speed = %d \n", DC\_speed);

}

STATE\_define(DC\_busy) {

DC\_state\_id = *DC\_busy*;

**printf**("US-------------- DC\_speed = %d -------------------------- \n",

DC\_speed);

}

