CHPDE

```
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"

/***MAIN***/
int main(int argc, char* argv[])
{
 //capture prog arguments.
 if(argc < 2){
   printf("Enter Filename ? \n");
   return 0;
 }
 printf("FILE -> %s\n", __FILE__);
 printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
 printf("DATE -> %d\n", __LINE__);
 printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
 printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
 /***Internal Variables***/
 int errno;
 int ires, ores;
 char *mem[lines][3]={
          {"1","um","one"},
          {"2","dois","two"},
          {"3","tres","three"},
          {"zero","0","zero"},
          {"zero","1","one"},
          {"one","0","one"},
          {"one","1","zero"}
 };
 /***USB FILE DESCRIPTOR***/

 /***nanosleep***/
 //alivea procesador.
 struct timespec* timer_1;
 timer_1 = calloc(1, sizeof(struct timespec));
 timer_1->tv_sec = 0;
 timer_1->tv_nsec = 100000;
 struct timespec* timer_2;
 timer_2 = calloc(1, sizeof(struct timespec));
 timer_2->tv_sec = 0;
 timer_2->tv_nsec = 100000;
 /******/
 /***Variables***/
```

```
  int i,j,n,KeyFound;
  //IN leitura;
  char keygen[2][word_size];
  char input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"zero");//output
  strcpy(keygen[1],"zero");//input
  strcpy(hist,"empty");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,word_size),word_size);
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*************************************************************************
***************/

    strcpy(ordem,MOME(mem,keygen,input));
    printf("ordem: %s\n",ordem);
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
```

```
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;
```

```
    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
```

```
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }
```

```
  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
```

```
//CS8     : 8n1 (8bit,no parity,1 stopbit)
//CLOCAL  : local connection, no modem control
//CREAD   : enable receiving characters

newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CREAD;

//ICANON  : enable canonical input
//disable all echo functionality, and don't send signals to calling program

//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
//in the comments, but we don't need them here

newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
newtio->c_cc[VERASE]   = 0;     // del -0
newtio->c_cc[VKILL]    = 0;     // @ -0
newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
```

```
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/*
```

```
//MOME from Matrix int
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//LOGIC from Matrix int
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//MOME from File
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
```

```
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from matrix
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
```

```
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
     break;
     }
   }//for iterator
   return keygen[0];
}
*/
/**/
//MOME from matrix
char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    KeyFound=!(strcmp(memory[iterator][0],keygen[0])||strcmp(memory[iterator][1],input));//bool
    printf("mem[0] %s mem[1] %s mem[2] %s\n",memory[iterator][0],memory[iterator]
[1],memory[iterator][2]);
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][2]);
            strcpy(keygen[1],input);
     break;
     }
   }//for iterator
   return keygen[0];
}
/**/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"

/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
```

```
printf("DATE -> %d\n", __LINE__);
printf("argv[0] (Progname) -> %s   size: %d          \n", argv[0], strlen(argv[0]));
printf("argv[1] (Filename)-> %s     size: %d          \n", argv[1], strlen(argv[1]));
/***Internal Variables***/
int errno;
int ires, ores;
/***USB FILE DESCRIPTOR***/
//int fd;
//struct termios oldtio,newtio;

//fd = open(DEVICE,  O_RDWR | O_NOCTTY | O_NDELAY);
//printf("file descriptor: %d\n",fd);
//if(fd < 0){
        //goto EXIT_2;
//}
//SetupSerial(fd,&oldtio,&newtio,BAUDRATE);
/***nanosleep***/
//alivea procesador.
struct timespec* timer_1;
timer_1 = calloc(1, sizeof(struct timespec));
timer_1->tv_sec = 0;
timer_1->tv_nsec = 100000;
struct timespec* timer_2;
timer_2 = calloc(1, sizeof(struct timespec));
timer_2->tv_sec = 0;
timer_2->tv_nsec = 100000;
/***Variables***/
int i,j,n,KeyFound;
//IN leitura;
char keygen[2][word_size];
char input[word_size];
char Input[word_size];
char hist[word_size];
char ordem[word_size];
/***File that stores de structs***/
FILE* fsm;
fsm=fopen(argv[1],"a+");
if(fsm == NULL)
  goto EXIT_1;
//saida ou estado inicial.
strcpy(keygen[0],"0");//output
strcpy(keygen[1],"0");//input
strcpy(hist,"empty");
perror("status ");
/***##########CICLOS MAQUINA############***/
for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
  //ENTRADAS.
  printf("Input: ");
  strncpy(input,ReadConsole(stdin,word_size),word_size);
```

```
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*****************************************************************************
***************/
    strcpy(Input,keygen[0]);
    strcat(Input,":");
    strcat(Input,input);
    printf("Input: %s\n",Input);

    LMOME(fsm,keygen,Input);
    strcpy(ordem,keygen[0]);
    printf("ordem: %s\n",ordem);
    //Write(fd,ordem,"\r\n");
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*****************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
  EXIT_2:
  printf("Premature Exiting\n");
  nanosleep(timer_1,NULL);
  //perror(DEVICE);
  //free(DEVICE);
  exit(-1);
  return -1;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
```

```
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
```

```
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;
```

```
}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
```

```
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;

  //ICANON  : enable canonical input
  //disable all echo functionality, and don't send signals to calling program
```

```
//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
//in the comments, but we don't need them here

newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
newtio->c_cc[VERASE]   = 0;     // del -0
newtio->c_cc[VKILL]    = 0;     // @ -0
newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
newtio->c_cc[VSWTC]    = 0;     // '\0' -0
newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
newtio->c_cc[VEOL]     = 0;     // '\0' -0
newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
newtio->c_cc[VEOL2]    = 0;     // '\0' -0
```

```
//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/*
//MOME from Matrix int
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
```

```
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//LOGIC from Matrix int
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//MOME from File
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
          //MOME Update
          strcpy(keygen[0],memory[2]);
          strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
```

```
}
*/
/*
//LOGIC from File
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from matrix
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//MOME from matrix
```

```
char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    KeyFound=!(strcmp(memory[iterator][0],keygen[0])||strcmp(memory[iterator][1],input));//bool
    printf("mem[0] %s mem[1] %s mem[2] %s\n",memory[iterator][0],memory[iterator]
[1],memory[iterator][2]);
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LMOME from matrix
char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/**/
//LOGIC from File
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
```

```
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/*
char *intostr(int value){
  int i;
  int temp=value;
  char *x;
  x=(char*)calloc(sizeof(char),12);
  for(i=0;temp!=0 && i<12;i++,temp/=10);
    x[i]='\0';i--;
  if(!(i<12))
    return NULL;
  for(temp=value,temp%=10;temp!=0;x[i]=(char)temp+48,value/=10,temp=value,i--,temp%=10);
  //printf("numero:- %s\n",x);
  return x;
}
*/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"

/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
```

```
printf("argv[0] (Progname) -> %s   size: %d        \n", argv[0], strlen(argv[0]));
printf("argv[1] (Filename)-> %s    size: %d        \n", argv[1], strlen(argv[1]));
/***Internal Variables***/
int errno;
int ires, ores;
/***USB FILE DESCRIPTOR***/
//int fd;
//struct termios oldtio,newtio;

//fd = open(DEVICE,  O_RDWR | O_NOCTTY | O_NDELAY);
//printf("file descriptor: %d\n",fd);
//if(fd < 0){
        //goto EXIT_2;
//}
//SetupSerial(fd,&oldtio,&newtio,BAUDRATE);
/***nanosleep***/
//alivea procesador.
struct timespec* timer_1;
timer_1 = calloc(1, sizeof(struct timespec));
timer_1->tv_sec = 0;
timer_1->tv_nsec = 100000;
struct timespec* timer_2;
timer_2 = calloc(1, sizeof(struct timespec));
timer_2->tv_sec = 0;
timer_2->tv_nsec = 100000;
/***Variables***/
//IN leitura;
char keygen[2][word_size];
char entrada[word_size];
char Input[word_size];
char hist[word_size];
char ordem[word_size];
/***File that stores de structs***/
FILE* fsm;
fsm=fopen(argv[1],"a+");
if(fsm == NULL)
  goto EXIT_1;
//saida ou estado inicial.
strcpy(keygen[0],"0");//output
strcpy(keygen[1],"0");//input
strcpy(hist,"empty");
perror("status ");

//printf("First Input: ");
/***##########CICLOS MAQUINA############***/
for(ires=0;TRUE;strcpy(hist,entrada)){
  //ENTRADAS.
  printf("Input: ");
  strcpy(entrada,ReadConsole(stdin,word_size));
```

```
    //entrada[strlen(entrada)]='\0';
    //printf("entrada: %s\n",entrada);
    /**/
    if(!strcmp(entrada,hist))//one shot
      continue;
    if(!strcmp(entrada,"exit")){
      goto EXIT_1;
    }
    /*********************************************************************************
***************/

    strcpy(Input,keygen[0]);
    strcat(Input,":");
    strcat(Input,entrada);
    //strcpy(Input,entrada);

    printf("Input: %s\n",Input);

    LMOME(fsm,keygen,Input,"-");

    strcpy(ordem,keygen[0]);
    printf("ordem:%s\n",ordem);
    //Write(fd,ordem,"\r\n");
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*********************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
  EXIT_2:
  printf("Premature Exiting\n");
  nanosleep(timer_1,NULL);
  //perror(DEVICE);
  //free(DEVICE);
  exit(-1);
  return -1;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"
```

```
/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;
```

```
}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;
```

```
    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
```

```
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;
```

```
//ICANON  : enable canonical input
//disable all echo functionality, and don't send signals to calling program

//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
//in the comments, but we don't need them here

newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
newtio->c_cc[VERASE]   = 0;     // del -0
newtio->c_cc[VKILL]    = 0;     // @ -0
newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
newtio->c_cc[VSWTC]    = 0;     // '\0' -0
newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
newtio->c_cc[VEOL]     = 0;     // '\0' -0
newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
```

```
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/*
//MOME from Matrix int
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
```

```
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//LOGIC from Matrix int
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//MOME from File
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
```

```
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from matrix
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
```

```
}
*/
/*
//MOME from matrix
char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    KeyFound=!(strcmp(memory[iterator][0],keygen[0])||strcmp(memory[iterator][1],input));//bool
    printf("mem[0] %s mem[1] %s mem[2] %s\n",memory[iterator][0],memory[iterator]
[1],memory[iterator][2]);
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LMOME from matrix
char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
```

```
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
char *intostr(int value){
  int i;
  int temp=value;
  char *x;
  x=(char*)calloc(sizeof(char),12);
  for(i=0;temp!=0 && i<12;i++,temp/=10);
    x[i]='\0';i--;
  if(!(i<12))
    return NULL;
  for(temp=value,temp%=10;temp!=0;x[i]=(char)temp+48,value/=10,temp=value,i--,temp%=10);
  //printf("numero:- %s\n",x);
  return x;
}
*/
/**/
//em fase de test
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size],char *parser)
{
  char line[2*word_size];
  char *token=NULL;
  char memory[2][word_size];
  int i;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];

  for(rewind(fp);fgets(line,word_size,fp);){
    line[strlen(line)-2]='\0';
```

```
   for(i=0,token=strtok(line,parser);token;i++,token=strtok(NULL,parser))
     strcpy(memory[i],token);

   if(!strcmp(memory[0],input)){
     printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
     //MOME Update
     strcpy(keygen[0],memory[1]);
     strcpy(keygen[1],input);
     break;
   }
  }//for iterator
  return keygen[0];
}
/**/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"

/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  int ires, ores;
  /***USB FILE DESCRIPTOR***/
  //int fd;
  //struct termios oldtio,newtio;

  //fd = open(DEVICE,  O_RDWR | O_NOCTTY | O_NDELAY);
  //printf("file descriptor: %d\n",fd);
  //if(fd < 0){
        //goto EXIT_2;
  //}
  //SetupSerial(fd,&oldtio,&newtio,BAUDRATE);
  /***nanosleep***/
```

```
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
  timer_1->tv_sec = 0;
  timer_1->tv_nsec = 100000;
  struct timespec* timer_2;
  timer_2 = calloc(1, sizeof(struct timespec));
  timer_2->tv_sec = 0;
  timer_2->tv_nsec = 100000;
  /***Variables***/
  //IN leitura;
  char keygen[2][word_size];
  char entrada[word_size];
  char Input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"empty");
  perror("status ");

  //printf("First Input: ");
  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,entrada)){
    //ENTRADAS.
    printf("Input: ");
    strcpy(entrada,ReadConsole(stdin,word_size));
    //entrada[strlen(entrada)]='\0';
    //printf("entrada: %s\n",entrada);
    /**/
    if(!strcmp(entrada,hist))//one shot
      continue;
    if(!strcmp(entrada,"exit")){
      goto EXIT_1;
    }
    /*************************************************************************************
***************/

    strcpy(Input,keygen[0]);
    strcat(Input,":");
    strcat(Input,entrada);
    //strcpy(Input,entrada);
```

```
    printf("Input: %s\n",Input);

    LMOME(fsm,keygen,Input,10,"-");

    strcpy(ordem,keygen[0]);
    printf("ordem:%s\n",ordem);
    //Write(fd,ordem,"\r\n");
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*************************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
  EXIT_2:
  printf("Premature Exiting\n");
  nanosleep(timer_1,NULL);
  //perror(DEVICE);
  //free(DEVICE);
  exit(-1);
  return -1;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;
```

```
  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
```

```
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;
```

```
}
/******/
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
```

```
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;

  //ICANON  : enable canonical input
  //disable all echo functionality, and don't send signals to calling program

  //newtio->c_lflag = ICANON | ECHOE;
  //newtio->c_lflag = ISIG;
  //newtio->c_lflag = ECHO;
  //newtio->c_lflag = ECHOK;
  //newtio->c_lflag = FLUSHO;
  //newtio->c_lflag = ICANON;
  newtio->c_lflag = ICANON | IEXTEN;
  //newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
  //newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
  //newtio->c_lflag = ICANON | IEXTEN | PENDIN;

  //IGNPAR  : ignore bytes with parity errors
```

```
  //ICRNL   : map CR to NL (otherwise a CR input on the other computer
  //          will not terminate input)
  //otherwise make device raw (no other input processing)

  //newtio->c_iflag |= (IGNPAR | ICRNL);
  //newtio->c_iflag |= (INPCK | ISTRIP);
  newtio->c_iflag |= INPCK;
  //newtio->c_iflag |= (INPCK | ICRNL);

  //Raw output. first option.

  //newtio->c_oflag |= 0;
  //newtio->c_oflag |= (OPOST | ONLCR);
  newtio->c_oflag |= OPOST;
  //newtio->c_oflag &= ~OPOST;

  //initialize all control characters
  //default values can be found in /usr/include/termios.h, and are given
  //in the comments, but we don't need them here

  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
```

```
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                    sent -> %s",ordem);
  free(ordem);
  return ores;
}
/*
//MOME from Matrix int
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//LOGIC from Matrix int
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
```

```
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//MOME from File
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
```

```
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from matrix
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//MOME from matrix
char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    KeyFound=!(strcmp(memory[iterator][0],keygen[0])||strcmp(memory[iterator][1],input));//bool
    printf("mem[0] %s mem[1] %s mem[2] %s\n",memory[iterator][0],memory[iterator]
[1],memory[iterator][2]);
    if(KeyFound){
            //MOME Update
```

```
            strcpy(keygen[0],memory[iterator][2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LMOME from matrix
char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
```

```
    }//for iterator
    return keygen[0];
}
*/
/*
char *intostr(int value){
  int i;
  int temp=value;
  char *x;
  x=(char*)calloc(sizeof(char),12);
  for(i=0;temp!=0 && i<12;i++,temp/=10);
    x[i]='\0';i--;
  if(!(i<12))
    return NULL;
  for(temp=value,temp%=10;temp!=0;x[i]=(char)temp+48,value/=10,temp=value,i--,temp%=10);
  //printf("numero:- %s\n",x);
  return x;
}
*/
/**/
//em fase de test
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size],int n_token,char *parser)
{
  int i;
  char *token[n_token];
  char line[2*word_size];

  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];

  for(rewind(fp);fgets(line,word_size,fp);){
    line[strlen(line)-2]='\0';

    for(i=0,token[i]=strtok(line,parser);token[i];i++,token[i]=strtok(NULL,parser)){
      //printf("%d tokens: %s\n",i,token[i]);
    }
    //Strtok(line,token,parser);


    if(!strcmp(token[0],input)){
      printf("mem[0]: %s mem[1]: %s\n",token[0],token[1]);
      //MOME Update
      strcpy(keygen[0],token[1]);
      strcpy(keygen[1],input);
      break;
    }
  }//for iterator
  return keygen[0];
}
```

```
/**/
/**/
char *Strtok(char *line,char *token[],char *parser)
{
  int i;
  for(i=0,token[i]=strtok(line,parser);token[i];i++,token[i]=strtok(NULL,parser)){
    //printf("%d tokens: %s \n",i,token[i]);
  }
  return line;
}
/**/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"

/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  //int ires, ores;
  /***USB FILE DESCRIPTOR***/
  int fd;
  struct termios oldtio,newtio;

  fd = open(DEVICE,  O_RDWR | O_NOCTTY | O_NDELAY);
  printf("file descriptor: %d\n",fd);
  if(fd < 0){
          goto EXIT_2;
  }
  SetupSerial(fd,&oldtio,&newtio,BAUDRATE);
  /***nanosleep***/
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
```

```
timer_1->tv_sec = 0;
timer_1->tv_nsec = 100000;
struct timespec* timer_2;
timer_2 = calloc(1, sizeof(struct timespec));
timer_2->tv_sec = 0;
timer_2->tv_nsec = 1000000000;
/***Variables***/
//IN leitura;

char keygen[2][word_size];
char entrada[word_size];
char Input[word_size];
char hist[word_size];
char ordem[word_size];
char *ptr[2];
//keygen=(char*)calloc(65,sizeof(char));
//ptr[1]=keygen;
/***File that stores de structs***/
FILE* fsm;
FILE* guicon;
fsm=fopen(argv[1],"a+");
if(fsm == NULL)
  goto EXIT_1;
guicon=fopen("guicon","r");
if(guicon == NULL)
  goto EXIT_1;
//saida ou estado inicial.
strcpy(keygen[0],"0");//input
strcpy(keygen[1],"0");//output
//keygen[0]=inic;
strcpy(hist,"0");
perror("status ");

//printf("First Input: ");
/***##########CICLOS MAQUINA############***/
for(printf("Start ");TRUE;strcpy(hist,entrada)){
//while(TRUE){strcpy(hist,entrada);
  //ENTRADAS.
  printf("Input: ");
  strcpy(entrada,ReadConsole(stdin,word_size));
  nanosleep(timer_2,NULL);//delay

  //rewind(guicon);
  //fgets(entrada,word_size,guicon);
  //entrada[strlen(entrada)-2]='\0';
  printf("entrada: %s\n",entrada);

  //entrada[strlen(entrada)]='\0';
  //printf("entrada: %s\n",entrada);
```

```
    /**/
    if(!strcmp(entrada,hist))//one shot
      continue;
    if(!strcmp(entrada,"exit")){
      goto EXIT_1;
    }
    /*****************************************************************************
***************/
    printf("keygen[1]:%s",keygen[1]);
    //strcpy(Input,hist);
    //strcat(Input,":");
    //strcpy(Input,keygen[1]);
    //strcat(Input,":");
    strcpy(Input,entrada);
    //strcpy(Input,entrada);

    printf("Input:%s\n",Input);

    LMOME(fsm,keygen,Input,"-");

    strcpy(ordem,keygen[1]);
    printf("ordem:%s\n",keygen[1]);
    Write(fd,ordem,"\r\n");
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*****************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  fclose(guicon);
  return 0;
  EXIT_2:
  printf("Premature Exiting\n");
  nanosleep(timer_1,NULL);
  exit(-1);
  return -1;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"
```

```
/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
```

```
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }
```

```
  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
```

```
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  //interanl struct variable
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;
```

```
//ICANON  : enable canonical input
//disable all echo functionality, and don't send signals to calling program

//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
//in the comments, but we don't need them here

newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
newtio->c_cc[VERASE]   = 0;     // del -0
newtio->c_cc[VKILL]    = 0;     // @ -0
newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
newtio->c_cc[VSWTC]    = 0;     // '\0' -0
newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
newtio->c_cc[VEOL]     = 0;     // '\0' -0
newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
```

```
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/*
//MOME from Matrix int
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
```

```
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//LOGIC from Matrix int
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
*/
/*
//MOME from File
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
```

```
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from matrix
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
```

```
}
*/
/*
//MOME from matrix
char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    KeyFound=!(strcmp(memory[iterator][0],keygen[0])||strcmp(memory[iterator][1],input));//bool
    printf("mem[0] %s mem[1] %s mem[2] %s\n",memory[iterator][0],memory[iterator]
[1],memory[iterator][2]);
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LMOME from matrix
char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
```

```
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
char *intostr(int value){
  int i;
  int temp=value;
  char *x;
  x=(char*)calloc(sizeof(char),12);
  for(i=0;temp!=0 && i<12;i++,temp/=10);
    x[i]='\0';i--;
  if(!(i<12))
    return NULL;
  for(temp=value,temp%=10;temp!=0;x[i]=(char)temp+48,value/=10,temp=value,i--,temp%=10);
  //printf("numero:- %s\n",x);
  return x;
}
*/
/**/
//em fase de test
char *LMOME(FILE *fp,char keygen[2][word_size],char *input,char *parser)
{
  int i;
  char line[2*word_size];
  char mem[2][word_size];
  char *token;
  for(rewind(fp);fgets(line,word_size,fp);){
    line[strlen(line)-2]='\0';

    for(i=0,token=strtok(line,parser);token;i++,token=strtok(NULL,parser)){
      if(i==0)
        strcpy(mem[0],token);
      if(i==1)
        strcpy(mem[1],token);
    }
    printf("mem[0] -> %s mem[1] -> %s\n",mem[0],mem[1]);
```

```
   if(!strcmp(mem[0],input)){
     //UPDATE
     strcpy(keygen[0],mem[0]);
     strcpy(keygen[1],mem[1]);
     break;
   }
  }//for iterator
  return keygen[1];
}
/**/
/**/
char *Strtok(char *line,char *token[],char *parser)
{
  int i;
  for(i=0,token[i]=strtok(line,parser);token[i];i++,token[i]=strtok(NULL,parser)){
    //printf("%d tokens: %s \n",i,token[i]);
  }
  return line;
}
/**/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"
/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  int ires, ores;
  /***USB FILE DESCRIPTOR***/

  /***nanosleep***/
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
```

```
  timer_1->tv_sec = 0;
  timer_1->tv_nsec = 100000;
  struct timespec* timer_2;
  timer_2 = calloc(1, sizeof(struct timespec));
  timer_2->tv_sec = 0;
  timer_2->tv_nsec = 100000;
  /******/
  /***Variables***/
  int i,j,n,KeyFound;
  //IN leitura;
  char keygen[2][word_size];
  char input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"empty");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,word_size),word_size);
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*****************************************************************************
***************/

    strcpy(ordem,LOGIC(fsm,keygen,input));
    printf("ordem: %s\n",ordem);
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*****************************************************************************
**/
  }//for TRUE
```

```
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
          perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
```

```
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
```

```
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
```

```
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;
```

```
//save old configuration and prepare newtio
tcgetattr(fd, oldtio);
bzero(newtio, sizeof(newtio));

//BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
//CRTSCTS : output hardware flow control (only used if the cable has
//          all necessary lines. See sect. 7 of Serial-HOWTO)
//CS8     : 8n1 (8bit,no parity,1 stopbit)
//CLOCAL  : local connection, no modem control
//CREAD   : enable receiving characters

newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CREAD;

//ICANON  : enable canonical input
//disable all echo functionality, and don't send signals to calling program

//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
```

```
  //in the comments, but we don't need them here

  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
```

```
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/****** MOME from Matrix******
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
//LOGIC from Matrix
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
***/
/***MOME from File***/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
```

```
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/***LOGIC from File***/
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/******/
/*
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
```

```
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"

/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  int ires, ores;
  char *mem[lines][2]={
            {"1","um"},
            {"2","dois"},
            {"3","tres"}
  };
  /***USB FILE DESCRIPTOR***/

  /***nanosleep***/
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
  timer_1->tv_sec = 0;
  timer_1->tv_nsec = 100000;
  struct timespec* timer_2;
  timer_2 = calloc(1, sizeof(struct timespec));
```

```
  timer_2->tv_sec = 0;
  timer_2->tv_nsec = 100000;
  /******/
  /***Variables***/
  int i,j,n,KeyFound;
  //IN leitura;
  char keygen[2][word_size];
  char input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"empty");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,word_size),word_size);
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*************************************************************************************
***************/

    strcpy(ordem,LOGIC(mem,keygen,input));
    printf("ordem: %s\n",ordem);
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*************************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
```

```
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
```

```
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
```

```
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
```

```
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));
```

```
//BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
//CRTSCTS : output hardware flow control (only used if the cable has
//          all necessary lines. See sect. 7 of Serial-HOWTO)
//CS8     : 8n1 (8bit,no parity,1 stopbit)
//CLOCAL  : local connection, no modem control
//CREAD   : enable receiving characters

newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CREAD;

//ICANON  : enable canonical input
//disable all echo functionality, and don't send signals to calling program

//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
//in the comments, but we don't need them here

newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
```

```
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
```

```
  free(ordem);
  return ores;
}
/****** MOME from Matrix******
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
//LOGIC from Matrix
int LOGIC(int mem[lines][2],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][1];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
***/
/*
//MOME from File
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s      %s\n",memory[0],memory[1],memory[2])!=EOF;){
```

```
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/*
//LOGIC from File
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[2][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s\n",memory[0],memory[1])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s\n",memory[0],memory[1]);
    KeyFound=!(strcmp(memory[0],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
*/
/**/
//LOGIC
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size])
{
  int iterator;
  int KeyFound;
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    printf("mem[0]: %s mem[1]: %s\n",memory[iterator][0],memory[iterator][1]);
    KeyFound=!(strcmp(memory[iterator][0],input));//bool
```

```
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[iterator][1]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/**/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
//this program is a learning finite state machine.

#include "creation_7.h"
/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter the board use duemilanove or mega or uno ? \n");
    return 0;
  }else if(argc < 3){
    printf("Enter a file name please \n");
    return 0;
  }else if(argc < 4){
    printf("Enter learning mode on or off \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("double -> %d\n", Double(5));
  //printf("Pow -> %f\n", Pow(2,5));
  printf("argv[0] (Progname) -> %s   size: %d        \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Board)-> %s       size: %d        \n", argv[1], strlen(argv[1]));
  printf("argv[2] (Filename)-> %s    size: %d        \n", argv[2], strlen(argv[2]));
  printf("argv[3] (Learnmode) -> %s size: %d        \n", argv[3], strlen(argv[3]));
  //make decisions relative income arguments.
  if(!strcmp(argv[1], "mega")){
    DEVICE = Putstr(DEVICE_1);
  }else if(!strcmp(argv[1], "duemilanove")){
    DEVICE = Putstr(DEVICE_2);
  }else if(!strcmp(argv[1], "uno")){
    DEVICE = Putstr(DEVICE_3);
  }else if(!strcmp(argv[1], "none")){
    DEVICE = Putstr(DEVICE_4);
```

```
}else{
  printf("Board options mega or duemilanove or uno or none\n");
  return 0;
}
if(!strcmp(argv[3], "on")){
  LEARN=1;
}else if(!strcmp(argv[3], "off")){
  LEARN=0;
}else{
  printf("LEARN mode options on or off\n");
  return 0;
}
/***Internal Variables***/
int errno;
int ires, ores;
/***USB FILE DESCRIPTOR***/
//int fd;
//struct termios oldtio,newtio;

//fd = open(DEVICE,  O_RDWR | O_NOCTTY | O_NDELAY);
//printf("file descriptor: %d\n",fd);
//if(fd < 0){
        //goto EXIT_2;
//}
//SetupSerial(fd,&oldtio,&newtio,BAUDRATE);
/*********************************************/
/***nanosleep***/
//alivea procesador.
struct timespec* timer_1;
timer_1 = calloc(1, sizeof(struct timespec));
timer_1->tv_sec = 0;
timer_1->tv_nsec = 100000;
struct timespec* timer_2;
timer_2 = calloc(1, sizeof(struct timespec));
timer_2->tv_sec = 0;
timer_2->tv_nsec = 100000;
/******/
/***Variables***/
int i,j,n,KeyFound;
//IN leitura;
char keygen[2][word_size];
//MEM rep;
//char memory[3][word_size];
char input[word_size];
char hist[word_size];
char ordem[word_size];
/***File that stores de structs***/
FILE* fsm;
fsm=fopen(argv[2],"a+");
```

```
  if(fsm == NULL)
    goto EXIT_1;
  //Inicialize the primary struct.
  //printf("struct size:%d\n\7",sizeof(leitura));
  //saida ou estado inicial.
  strcpy(keygen[0],"0");
  strcpy(keygen[1],"0");
  strcpy(hist,"empty");
  //delay.
  printf("    Press reset button on target!!\n");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,buf_size),word_size);
    if(!(strlen(input) < word_size))
                    input[word_size-1]='\0';

    /*
    //Entrada Serial
    for(nanosleep(timer_1,NULL);!read(fd,leitura.key[2],buf_size);nanosleep(timer_1,NULL));

    for(ires=0;ires < word_size;ires++){
      if(leitura.key[2][ires] == '\r' || leitura.key[2][ires] == '\n'){
           leitura.key[2][ires] = '\0';
           break;
      }
    }
    */


    //if(!strcmp(input,hist))//one shot
      //continue;
    printf("\ninput : %s\n",input);
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*************************************************************************
***************/


    strcpy(ordem,MOME(fsm,keygen,input));

    printf("ordem: %s keygen[0]: %s keygen[1]: %s input: %s\n",ordem,keygen[0],keygen[1],input);


  /*************************************************************************
```

```
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  /* restore the old port settings */
  //tcsetattr(fd, TCSANOW, &oldtio);
  //close(fd);
  free(DEVICE);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
  EXIT_2:
  printf("Premature Exiting\n");
  nanosleep(timer_1,NULL);
  perror(DEVICE);
  free(DEVICE);
  exit(-1);
  return -1;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
```

```
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting
```

```
    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
```

```
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
```

```
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;

  //ICANON  : enable canonical input
  //disable all echo functionality, and don't send signals to calling program

  //newtio->c_lflag = ICANON | ECHOE;
  //newtio->c_lflag = ISIG;
  //newtio->c_lflag = ECHO;
  //newtio->c_lflag = ECHOK;
  //newtio->c_lflag = FLUSHO;
  //newtio->c_lflag = ICANON;
  newtio->c_lflag = ICANON | IEXTEN;
  //newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
  //newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
  //newtio->c_lflag = ICANON | IEXTEN | PENDIN;

  //IGNPAR  : ignore bytes with parity errors
  //ICRNL   : map CR to NL (otherwise a CR input on the other computer
  //          will not terminate input)
  //otherwise make device raw (no other input processing)

  //newtio->c_iflag |= (IGNPAR | ICRNL);
```

```
  //newtio->c_iflag |= (INPCK | ISTRIP);
  newtio->c_iflag |= INPCK;
  //newtio->c_iflag |= (INPCK | ICRNL);

  //Raw output. first option.

  //newtio->c_oflag |= 0;
  //newtio->c_oflag |= (OPOST | ONLCR);
  newtio->c_oflag |= OPOST;
  //newtio->c_oflag &= ~OPOST;

  //initialize all control characters
  //default values can be found in /usr/include/termios.h, and are given
  //in the comments, but we don't need them here

  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
```

```
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                    sent -> %s",ordem);
  free(ordem);
  return ores;
}
/******
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
***/
/******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
```

```
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/******/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
//this program is a learning finite state machine.

#include "creation_7.h"
/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter the board use duemilanove or mega or uno ? \n");
    return 0;
  }else if(argc < 3){
    printf("Enter a file name please \n");
    return 0;
  }else if(argc < 4){
    printf("Enter learning mode on or off \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("double -> %d\n", Double(5));
  //printf("Pow -> %f\n", Pow(2,5));
  printf("argv[0] (Progname) -> %s   size: %d        \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Board)-> %s       size: %d        \n", argv[1], strlen(argv[1]));
  printf("argv[2] (Filename)-> %s    size: %d        \n", argv[2], strlen(argv[2]));
  printf("argv[3] (Learnmode) -> %s size: %d        \n", argv[3], strlen(argv[3]));
  //make decisions relative income arguments.
  if(!strcmp(argv[1], "mega")){
    DEVICE = Putstr(DEVICE_1);
  }else if(!strcmp(argv[1], "duemilanove")){
    DEVICE = Putstr(DEVICE_2);
  }else if(!strcmp(argv[1], "uno")){
    DEVICE = Putstr(DEVICE_3);
```

```
}else if(!strcmp(argv[1], "none")){
  DEVICE = Putstr(DEVICE_4);
}else{
  printf("Board options mega or duemilanove or uno or none\n");
  return 0;
}
if(!strcmp(argv[3], "on")){
  LEARN=1;
}else if(!strcmp(argv[3], "off")){
  LEARN=0;
}else{
  printf("LEARN mode options on or off\n");
  return 0;
}
/***Internal Variables***/
int errno;
int ires, ores;
/***USB FILE DESCRIPTOR***/
int fd;
struct termios oldtio,newtio;

fd = open(DEVICE,  O_RDWR | O_NOCTTY | O_NDELAY);
printf("file descriptor: %d\n",fd);
if(fd < 0){
        goto EXIT_2;
}
SetupSerial(fd,&oldtio,&newtio,BAUDRATE);
/*****************************************/
/***nanosleep***/
//alivea procesador.
struct timespec* timer_1;
timer_1 = calloc(1, sizeof(struct timespec));
timer_1->tv_sec = 0;
timer_1->tv_nsec = 100000;
struct timespec* timer_2;
timer_2 = calloc(1, sizeof(struct timespec));
timer_2->tv_sec = 0;
timer_2->tv_nsec = 100000;
/******/
/***Variables***/
int i,j,n,KeyFound;
//IN leitura;
char keygen[2][word_size];
//MEM rep;
//char memory[3][word_size];
char input[word_size];
char hist[word_size];
char ordem[word_size];
/***File that stores de structs***/
```

```
  FILE* fsm;
  fsm=fopen(argv[2],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //Inicialize the primary struct.
  //printf("struct size:%d\n\7",sizeof(leitura));
  //saida ou estado inicial.
  strcpy(keygen[0],"0");
  strcpy(keygen[1],"0");
  strcpy(hist,"empty");
  //delay.
  printf("    Press reset button on target!!\n");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    //printf("Input: ");

    //strncpy(input,ReadConsole(stdin,buf_size),word_size);
    //if(!(strlen(input) < word_size))
                    //input[word_size-1]='\0';

    /**/
    //Entrada Serial
    ires=read(fd,input,word_size);
    for(nanosleep(timer_1,NULL);!ires;nanosleep(timer_1,NULL));

    for(ires=0;ires < word_size;ires++){
      if(input[ires] == '\r' || input[ires] == '\n'){
          input[ires] = '\0';
          break;
      }
    }
    /**/


    if(!strcmp(input,hist))//one shot
      continue;
    printf("\ninput : %s\n",input);
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*************************************************************************
**************/


    strcpy(ordem,MOME(fsm,keygen,input));
```

```
    if(!strcmp("exit",ordem))
      goto EXIT_1;

    ores=Write(fd,ordem,"\r\n");

   /*****************************************************************************
**/
   }//for TRUE
   //EXITS
   EXIT_1:
   printf("Exiting\n");
   nanosleep(timer_1,NULL);
   /* restore the old port settings */
   tcsetattr(fd, TCSANOW, &oldtio);
   close(fd);
   free(DEVICE);
   free(timer_1);
   free(timer_2);
   fclose(fsm);
   return 0;
   EXIT_2:
   printf("Premature Exiting\n");
   nanosleep(timer_1,NULL);
   perror(DEVICE);
   free(DEVICE);
   exit(-1);
   return -1;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
   int i; char* ptr;
   ptr = (char*)calloc(strlen(str), sizeof(char));
   if(ptr == NULL){
     perror("NULL!\n");
     return NULL;

   }
   for(i=0; (ptr[i] = str[i]); i++){
     if(ptr[i] == '\0')
       break;

   }
```

```
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
```

```
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;
```

```
}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
```

```
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;

  //ICANON  : enable canonical input
  //disable all echo functionality, and don't send signals to calling program

  //newtio->c_lflag = ICANON | ECHOE;
  //newtio->c_lflag = ISIG;
  //newtio->c_lflag = ECHO;
  //newtio->c_lflag = ECHOK;
  //newtio->c_lflag = FLUSHO;
  //newtio->c_lflag = ICANON;
  newtio->c_lflag = ICANON | IEXTEN;
  //newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
  //newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
  //newtio->c_lflag = ICANON | IEXTEN | PENDIN;
```

```
  //IGNPAR  : ignore bytes with parity errors
  //ICRNL   : map CR to NL (otherwise a CR input on the other computer
  //          will not terminate input)
  //otherwise make device raw (no other input processing)

  //newtio->c_iflag |= (IGNPAR | ICRNL);
  //newtio->c_iflag |= (INPCK | ISTRIP);
  newtio->c_iflag |= INPCK;
  //newtio->c_iflag |= (INPCK | ICRNL);

  //Raw output. first option.

  //newtio->c_oflag |= 0;
  //newtio->c_oflag |= (OPOST | ONLCR);
  newtio->c_oflag |= OPOST;
  //newtio->c_oflag &= ~OPOST;

  //initialize all control characters
  //default values can be found in /usr/include/termios.h, and are given
  //in the comments, but we don't need them here

  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
   }
  return 0;
}
/***ReadInt***/
```

```
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                    sent -> %s",ordem);
  free(ordem);
  return ores;
}
/******
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
***/
/******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
```

```
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s      %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/******/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
//this program is a learning finite state machine.

#include "creation_7.h"
/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  int ires, ores;
  /***USB FILE DESCRIPTOR***/

  /***nanosleep***/
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
  timer_1->tv_sec = 0;
  timer_1->tv_nsec = 100000;
```

```
  struct timespec* timer_2;
  timer_2 = calloc(1, sizeof(struct timespec));
  timer_2->tv_sec = 0;
  timer_2->tv_nsec = 100000;
  /******/
  /***Variables***/
  int i,j,n,KeyFound;
  //IN leitura;
  char keygen[2][word_size];
  char input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"empty");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,word_size),word_size);
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*****************************************************************************
***************/

    strcpy(ordem,MOME(fsm,keygen,input));
    printf("ordem: %s\n",ordem);
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*****************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
```

```
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;
```

```
    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
```

```
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
```

```
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
```

```
tcgetattr(fd, oldtio);
bzero(newtio, sizeof(newtio));

//BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
//CRTSCTS : output hardware flow control (only used if the cable has
//          all necessary lines. See sect. 7 of Serial-HOWTO)
//CS8     : 8n1 (8bit,no parity,1 stopbit)
//CLOCAL  : local connection, no modem control
//CREAD   : enable receiving characters

newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
//newtio->c_cflag = speed | CS8 | CREAD;

//ICANON  : enable canonical input
//disable all echo functionality, and don't send signals to calling program

//newtio->c_lflag = ICANON | ECHOE;
//newtio->c_lflag = ISIG;
//newtio->c_lflag = ECHO;
//newtio->c_lflag = ECHOK;
//newtio->c_lflag = FLUSHO;
//newtio->c_lflag = ICANON;
newtio->c_lflag = ICANON | IEXTEN;
//newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
//newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
//newtio->c_lflag = ICANON | IEXTEN | PENDIN;

//IGNPAR  : ignore bytes with parity errors
//ICRNL   : map CR to NL (otherwise a CR input on the other computer
//          will not terminate input)
//otherwise make device raw (no other input processing)

//newtio->c_iflag |= (IGNPAR | ICRNL);
//newtio->c_iflag |= (INPCK | ISTRIP);
newtio->c_iflag |= INPCK;
//newtio->c_iflag |= (INPCK | ICRNL);

//Raw output. first option.

//newtio->c_oflag |= 0;
//newtio->c_oflag |= (OPOST | ONLCR);
newtio->c_oflag |= OPOST;
//newtio->c_oflag &= ~OPOST;

//initialize all control characters
//default values can be found in /usr/include/termios.h, and are given
//in the comments, but we don't need them here
```

```
  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
```

```
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/******
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
***/
/******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/******/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
```

```
/***CABECALHO Libraries***/
//this program is a learning finite state machine.

#include "creation_7.h"
/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  int ires, ores;
  /***USB FILE DESCRIPTOR***/

  /***nanosleep***/
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
  timer_1->tv_sec = 0;
  timer_1->tv_nsec = 100000;
  struct timespec* timer_2;
  timer_2 = calloc(1, sizeof(struct timespec));
  timer_2->tv_sec = 0;
  timer_2->tv_nsec = 100000;
  /******/
  /***Variables***/
  int i,j,n,KeyFound;
  //IN leitura;
  char keygen[2][word_size];
  char input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"empty");
```

```
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,word_size),word_size);
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*********************************************************************************
***************/

    strcpy(ordem,MOME(fsm,keygen,input));
    printf("ordem: %s\n",ordem);
    if(!strcmp("exit",ordem))
      goto EXIT_1;

  /*********************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;
```

```
  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;
```

```
}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;

}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
```

```
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
           perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\r';
      i++;
      *(value+i)='\n';
      i++;
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnumber***/
int getnumber(char* x)
{
  int num;
  if(sscanf(x, "%d", &num))
    return num;
  else
    return 0;

}
```

```
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
  int n;
  char* a;
  a=calloc(1024, sizeof(char));
  while((n=fscanf(stream, "%s", a))){
    if(strlen(a) > (Size_Inf-3)){
      printf("overflow retry.\n");
      continue;

    }
    strncpy(inf, a, (Size_Inf-3));
    strcat(inf, "\r\n");
    break;

  }
  free(a);
  return n;

}
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;

  //ICANON  : enable canonical input
  //disable all echo functionality, and don't send signals to calling program

  //newtio->c_lflag = ICANON | ECHOE;
  //newtio->c_lflag = ISIG;
  //newtio->c_lflag = ECHO;
  //newtio->c_lflag = ECHOK;
```

```
  //newtio->c_lflag = FLUSHO;
  //newtio->c_lflag = ICANON;
  newtio->c_lflag = ICANON | IEXTEN;
  //newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
  //newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
  //newtio->c_lflag = ICANON | IEXTEN | PENDIN;

  //IGNPAR  : ignore bytes with parity errors
  //ICRNL   : map CR to NL (otherwise a CR input on the other computer
  //          will not terminate input)
  //otherwise make device raw (no other input processing)

  //newtio->c_iflag |= (IGNPAR | ICRNL);
  //newtio->c_iflag |= (INPCK | ISTRIP);
  newtio->c_iflag |= INPCK;
  //newtio->c_iflag |= (INPCK | ICRNL);

  //Raw output. first option.

  //newtio->c_oflag |= 0;
  //newtio->c_oflag |= (OPOST | ONLCR);
  newtio->c_oflag |= OPOST;
  //newtio->c_oflag &= ~OPOST;

  //initialize all control characters
  //default values can be found in /usr/include/termios.h, and are given
  //in the comments, but we don't need them here

  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
```

```
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                    sent -> %s",ordem);
  free(ordem);
  return ores;
}
/******
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
```

```
  return keygen[0];
}
***/
/******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
      }
    }//for iterator
    return keygen[0];
}
/******/
/***sergio manuel salazar dos santos***/
/***Tel:- 916919898***/
/***Rua do relogio 268, 4770-245 Joane, Vila Nova de famalicao, Braga, Portugal***/
/***CABECALHO Libraries***/
/*Moore Mealy*/

#include "creation_7.h"
/***MAIN***/
int main(int argc, char* argv[])
{
  //capture prog arguments.
  if(argc < 2){
    printf("Enter Filename ? \n");
    return 0;
  }
  printf("FILE -> %s\n", __FILE__);
  printf("DATE -> %s TIME -> %s\n", __DATE__, __TIME__);
  printf("DATE -> %d\n", __LINE__);
  printf("argv[0] (Progname) -> %s   size: %d         \n", argv[0], strlen(argv[0]));
  printf("argv[1] (Filename)-> %s     size: %d         \n", argv[1], strlen(argv[1]));
  /***Internal Variables***/
  int errno;
  int ires, ores;
  /***USB FILE DESCRIPTOR***/
```

```
  /***nanosleep***/
  //alivea procesador.
  struct timespec* timer_1;
  timer_1 = calloc(1, sizeof(struct timespec));
  timer_1->tv_sec = 0;
  timer_1->tv_nsec = 100000;
  struct timespec* timer_2;
  timer_2 = calloc(1, sizeof(struct timespec));
  timer_2->tv_sec = 0;
  timer_2->tv_nsec = 100000;
  /******/
  /***Variables***/
  int i,j,n,KeyFound;
  //IN leitura;
  char keygen[2][word_size];
  char input[word_size];
  char hist[word_size];
  char ordem[word_size];
  /***File that stores de structs***/
  FILE* fsm;
  fsm=fopen(argv[1],"a+");
  if(fsm == NULL)
    goto EXIT_1;
  //saida ou estado inicial.
  strcpy(keygen[0],"0");//output
  strcpy(keygen[1],"0");//input
  strcpy(hist,"empty");
  perror("status ");

  /***##########CICLOS MAQUINA############***/
  for(ires=0;TRUE;strcpy(hist,input),KeyFound=0){
    //ENTRADAS.
    printf("Input: ");
    strncpy(input,ReadConsole(stdin,word_size),word_size);
    if(!(strlen(input) < word_size))
      input[word_size-1]='\0';
    /**/
    if(!strcmp(input,hist))//one shot
      continue;
    if(!strcmp(input,"exit")){
      goto EXIT_1;
    }
    /*********************************************************************************
***************/

    strcpy(ordem,MOME(fsm,keygen,input));
    printf("ordem: %s\n",ordem);
    if(!strcmp("exit",ordem))
```

```
    goto EXIT_1;

  /*****************************************************************************
**/
  }//for TRUE
  //EXITS
  EXIT_1:
  printf("Exiting\n");
  nanosleep(timer_1,NULL);
  free(timer_1);
  free(timer_2);
  fclose(fsm);
  return 0;
}//main
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/

#include "creation_7.h"

/*FUNCOES*/
/*****Putstr*******/
char* Putstr(char* str)
{
  int i; char* ptr;
  ptr = (char*)calloc(strlen(str), sizeof(char));
  if(ptr == NULL){
    perror("NULL!\n");
    return NULL;

  }
  for(i=0; (ptr[i] = str[i]); i++){
    if(ptr[i] == '\0')
      break;

  }
  return (ptr);

}
/***ReadConsole***/
char *ReadConsole(FILE* stream,size_t n)
{
  int i, NBytes;
  char caracter;
  char *value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF; i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value, NBytes*sizeof(char));
      if(value==NULL)
```

```
            perror(value);

    }
    *(value+i)=caracter;
    if(caracter=='\n'){
      *(value+i)='\0';
      break;

    }
    if(i == n){
      *(value+i)='\0';
      break;

    }

  }
  return value;

}
/***getnum***/
int getnum(int min, int max)
{
  int num;
  for(num=0; !scanf("%d",&num) || num<min || num>max ; getchar()){
    perror("loop status");

  }
  return num;

}

/**********fillvec*************/
int* fillvec(int num, int size)
{
  int* x;
  int i;
  if(size > 0){
    x=calloc(size, sizeof(int));
    for(i=0; i<=size; i++){
      x[i]=num;
      //printf("x[%d]-> %d\n",i,x[i]);//troubleshooting

    }

  }else{
    return NULL;

  }
  return x;
```

```
}
/*******ReadFiletoMem*******/
void* ReadFiletoMem(void* datatype, FILE* filename)
{
  int i , n;
  fseek(filename, 0, SEEK_SET);
  datatype=calloc(1,sizeof(*datatype));
  for(i=0, n=1; fread((datatype+i), sizeof(*datatype), 1, filename); datatype=realloc(datatype,
n*sizeof(*datatype))){
    i++;
    n++;
    if(feof(filename))
      break;
    if(ferror(filename)){
      perror("status:");
      return NULL;

    }

  }
  return datatype;

}
/**GetChar()**/
unsigned char GetChar()
{
  unsigned char x;
  unsigned char value;
  for(value=getchar(); x!='\n'; x=getchar());
  if(value=='\0')
    return 1;
  x='0';
  return value;

}
/******/
//sintaxe muito muito importante, mais importante doque a semantica.
//I learn allot reading other peoples code.
char* ReadConsoleSer(FILE* stream)
{
  int i, NBytes;
  char caracter;
  char* value=NULL;
  for(i=0, NBytes=8; (caracter=getc(stream)) != EOF;i++){
    if((i==NBytes) | (i==0)){
      NBytes=2*NBytes;
      value=(char*)realloc(value,NBytes*sizeof(char)+2);
      if(value==NULL)
```

```
            perror(value);

      }
      *(value+i)=caracter;
      if(caracter=='\n'){
        *(value+i)='\r';
        i++;
        *(value+i)='\n';
        i++;
        *(value+i)='\0';
        break;

      }

    }
    return value;

}
/***getnumber***/
int getnumber(char* x)
{
    int num;
    if(sscanf(x, "%d", &num))
      return num;
    else
      return 0;

}
/***infor***/
int infor(char* inf, int Size_Inf, FILE* stream)
{
    int n;
    char* a;
    a=calloc(1024, sizeof(char));
    while((n=fscanf(stream, "%s", a))){
      if(strlen(a) > (Size_Inf-3)){
        printf("overflow retry.\n");
        continue;

      }
      strncpy(inf, a, (Size_Inf-3));
      strcat(inf, "\r\n");
      break;

    }
    free(a);
    return n;

}
```

```
/***SetupSerial***/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed)
{
  //portcomunication setup file descriptor
  int c;

  //save old configuration and prepare newtio
  tcgetattr(fd, oldtio);
  bzero(newtio, sizeof(newtio));

  //BAUDRATE: Set bps rate. You could also use cfsetispeed and cfsetospeed.
  //CRTSCTS : output hardware flow control (only used if the cable has
  //          all necessary lines. See sect. 7 of Serial-HOWTO)
  //CS8     : 8n1 (8bit,no parity,1 stopbit)
  //CLOCAL  : local connection, no modem control
  //CREAD   : enable receiving characters

  newtio->c_cflag = speed | CRTSCTS | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CLOCAL | CREAD;
  //newtio->c_cflag = speed | CS8 | CREAD;

  //ICANON  : enable canonical input
  //disable all echo functionality, and don't send signals to calling program

  //newtio->c_lflag = ICANON | ECHOE;
  //newtio->c_lflag = ISIG;
  //newtio->c_lflag = ECHO;
  //newtio->c_lflag = ECHOK;
  //newtio->c_lflag = FLUSHO;
  //newtio->c_lflag = ICANON;
  newtio->c_lflag = ICANON | IEXTEN;
  //newtio->c_lflag = ICANON | IEXTEN | FLUSHO;
  //newtio->c_lflag = ICANON | IEXTEN | ECHOKE;
  //newtio->c_lflag = ICANON | IEXTEN | PENDIN;

  //IGNPAR  : ignore bytes with parity errors
  //ICRNL   : map CR to NL (otherwise a CR input on the other computer
  //          will not terminate input)
  //otherwise make device raw (no other input processing)

  //newtio->c_iflag |= (IGNPAR | ICRNL);
  //newtio->c_iflag |= (INPCK | ISTRIP);
  newtio->c_iflag |= INPCK;
  //newtio->c_iflag |= (INPCK | ICRNL);

  //Raw output. first option.

  //newtio->c_oflag |= 0;
  //newtio->c_oflag |= (OPOST | ONLCR);
```

```
  newtio->c_oflag |= OPOST;
  //newtio->c_oflag &= ~OPOST;

  //initialize all control characters
  //default values can be found in /usr/include/termios.h, and are given
  //in the comments, but we don't need them here

  newtio->c_cc[VINTR]    = 0;     // Ctrl-c -0
  newtio->c_cc[VQUIT]    = 0;     // Ctrl-\ -0
  newtio->c_cc[VERASE]   = 0;     // del -0
  newtio->c_cc[VKILL]    = 0;     // @ -0
  newtio->c_cc[VEOF]     = 4;    // Ctrl-d -4
  newtio->c_cc[VTIME]    = 0;     // inter-character timer unused -0
  newtio->c_cc[VMIN]     = 1;     // blocking read until 1 character arrives -1
  newtio->c_cc[VSWTC]    = 0;     // '\0' -0
  newtio->c_cc[VSTART]   = 0;     // Ctrl-q -0
  newtio->c_cc[VSTOP]    = 0;     // Ctrl-s -0
  newtio->c_cc[VSUSP]    = 0;     // Ctrl-z -0
  newtio->c_cc[VEOL]     = 0;     // '\0' -0
  newtio->c_cc[VREPRINT] = 0;     // Ctrl-r -0
  newtio->c_cc[VDISCARD] = 0;     // Ctrl-u -0
  newtio->c_cc[VWERASE]  = 0;     // Ctrl-w -0
  newtio->c_cc[VLNEXT]   = 0;     // Ctrl-v -0
  newtio->c_cc[VEOL2]    = 0;     // '\0' -0

//now clean the modem line and activate the settings for the port
  tcflush(fd, TCIFLUSH);
  if(tcsetattr(fd, TCSANOW, newtio) != 0){
    tcsetattr(fd, TCSANOW, oldtio);
    close(fd);
    return -1;
  }
  return 0;
}
/***ReadInt***/
int ReadInt(int nmin, int nmax)
{
  int num;
  int flag;
  for(flag=1; flag;){
    for( num=0; !scanf("%d",&num); getchar());
    //printf("num: %d nmin: %d nmax: %d\n",num, nmin, nmax);
    if((num < nmin) || (num > nmax))
      continue;
    flag=0;
    }
  return num;
  }
/***Write***/
```

```
int Write(int fd,char *string,char *end)
{
  int ores;
  char *ordem;
  ordem=(char *)calloc(strlen(string)+strlen(end),sizeof(char));
  strcpy(ordem,string);
  strcat(ordem,end);
  ores = write(fd, ordem, strlen(ordem));
  printf("                  sent -> %s",ordem);
  free(ordem);
  return ores;
}
/****** MOME from Matrix
int MOME(int mem[lines][3],int keygen[2],int input)
{
  int iterator;
  int keyfound;
  if(keygen[1]==input)//previne redundancia.
    return keygen[0];
  for(iterator=0;iterator<lines;iterator++){
    keyfound=(mem[iterator][0]==keygen[0] && mem[iterator][1]==input);//bool
    if(keyfound){
      //MOME UPDATE
      keygen[0]=mem[iterator][2];
      keygen[1]=input;
      break;
    }
  }//for iterator
  return keygen[0];
}
***/
/***MOME from File***/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size])
{
  int KeyFound;
  char memory[3][word_size];
  if(!strcmp(keygen[1],input))//evitar redundancia
    return keygen[0];
  for(rewind(fp);fscanf(fp,"%s          %s       %s\n",memory[0],memory[1],memory[2])!=EOF;){
    if(ferror(fp)){
      perror("status :"); errno = 0; break;
    }
    printf("mem[0]: %s mem[1]: %s mem[2]: %s\n",memory[0],memory[1],memory[2]);
    KeyFound=!(strcmp(memory[0],keygen[0])||strcmp(memory[1],input));//bool
    if(KeyFound){
            //MOME Update
            strcpy(keygen[0],memory[2]);
            strcpy(keygen[1],input);
      break;
```

```
      }
    }//for iterator
    return keygen[0];
}
/******/
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
```

```
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define lines 7
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
```

```
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
//char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
```

```
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define DEVICE "/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Uno_649353431333512121A0-if00"
#define BAUDRATE B9600
#define TRUE 1
#define FALSE 0
#define lines 4
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
```

```
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
//char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
//char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size]);
/*******/
//char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *intostr(int value);
/*******/

#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
```

```
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define DEVICE "/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Uno_649353431333512121A0-if00"
#define BAUDRATE B9600
#define TRUE 1
#define FALSE 0
#define lines 4
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
```

```
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
//char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
//char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size]);
/*******/
//char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size],char *parser);
/*******/
//char *intostr(int value);
/*******/

#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
```

```
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
```

```
#define DEVICE "/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Uno_649353431333512121A0-if00"
#define BAUDRATE B9600
#define TRUE 1
#define FALSE 0
#define lines 4
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
//char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
```

```
//char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
//char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size]);
/*******/
//char *intostr(int value);
/*******/
char *Strtok(char *line,char *token[],char *parser);
/*******/
//char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
char *LMOME(FILE *fp,char keygen[2][word_size],char input[word_size],int n_token,char *parser);
/*******/

#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
```

```
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define DEVICE "/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Uno_649353431333512121A0-if00"
#define BAUDRATE B9600
#define TRUE 1
#define FALSE 0
#define lines 4
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
```

```
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
//char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
//char *MOME(char *memory[lines][3],char keygen[2][word_size],char input[word_size]);
/*******/
//char *intostr(int value);
/*******/
char *Strtok(char *line,char *token[],char *parser);
/*******/
//char *LMOME(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
char *LMOME(FILE *fp,char keygen[2][word_size],char *input,char *parser);
/*******/

#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
```

```
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
```

```
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
```

```
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define lines 3
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
```

```
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
//int MOME(int mem[lines][3],int keygen[2],int input);
/*******/
//int LOGIC(int mem[lines][2],int keygen[2],int input);
/*******/
//char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
//char *LOGIC(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
char *LOGIC(char *memory[lines][2],char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
```

```
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define BAUDRATE B9600
//arduino MEGA
#define DEVICE_1 "/dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A600aiS5-if00-port0"
//arduino DUEMILANOVE
#define DEVICE_2 "/dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A9007Qwg-if00-port0"
```

```
//arduino uno
#define DEVICE_3 "/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Uno_649353431333512121A0-if00"
//open free tty
#define DEVICE_4 "/home/sergio/fd.txt"
#define _POSIX_SOURCE 1
#define word_size 65
#define buf_size 195
#define Double(x) (2*(x))
/***Gloabal Variables***/
int LEARN;
char* DEVICE;
/***PROTO***/
//typedef union {
//  char memory[3][buf_size];
//}MEM;
//typedef union {
//  char keygen[2][word_size];
//}IN;
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
```

```
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
```

```
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define BAUDRATE B9600
//arduino MEGA
#define DEVICE_1 "/dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A600aiS5-if00-port0"
//arduino DUEMILANOVE
#define DEVICE_2 "/dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A9007Qwg-if00-port0"
//arduino uno
#define DEVICE_3 "/dev/serial/by-id/usb-
Arduino__www.arduino.cc__Arduino_Uno_649353431333512121A0-if00"
//open free tty
#define DEVICE_4 "/home/sergio/fd.txt"
#define _POSIX_SOURCE 1
#define word_size 65
#define buf_size 195
#define Double(x) (2*(x))
/***Gloabal Variables***/
int LEARN;
char* DEVICE;
/***PROTO***/
//typedef union {
//  char memory[3][buf_size];
//}MEM;
//typedef union {
//  char keygen[2][word_size];
//}IN;
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
```

```
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
```

```
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
```

```
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
```

```
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
```

```
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
#endif
/*creation_7.h*/
#ifdef _CREATION_7_H_
#else
#define _CREATION_7_H_
//PROTOTYPE
/**sergio manuel salazar dos santos 916919898**/
/***Libraries***/
// fopen perror fread fwrite feof fseek ferror fclose rewind scanf sscanf getchar scanf fscanf
// strncpy sscanf
#include <stdio.h>
// calloc free realloc
#include <stdlib.h>
// strcpy strcmp strcat memcmp
#include <string.h>
// termios tcflush
#include <termios.h>
// nanosleep sleep
#include <time.h>
// tcflsuh read write close
#include <unistd.h>
// perror
#include <errno.h>
// open
#include <sys/types.h>
#include <sys/stat.h>
//#include <sys/dir.h>
//#include <syscalls.h>
#include <fcntl.h>
//assert
#include <assert.h>
//offsetof
#include <stddef.h>
//__fpurge
//#include <stdio_ext.h>
```

```
//#include <netdb.h>
//#include <netinet/in.h>
//#include <stdbool.h>
//#include <ctype.h>
//#include <limits.h>
//#include <semaphore.h>
//#include <pthread.h>
//#include <math.h>
//#include <signal.h>
//#include <sys/mman.h>
//#include <sys/wait.h>
//#include <sys/time.h>
//#include <sys/resource.h>
#include <sys/dir.h>
//#include <sys/socket.h>
//#include <sys/un.h>
#include <sys/ioctl.h>
/***MACROS***/
#define TRUE 1
#define FALSE 0
#define _POSIX_SOURCE 1
#define word_size 65
#define Double(x) (2*(x))
#endif
/***FUNCTION TITLES***/
#include "creation_7_func.h"
/*creation_7_func.h*/
#ifdef _CREATION_7_FUNC_H_
#else
#define _CREATION_7_FUNC_H
//PROTOTYPE
/***FUNCTION TITLES***/
/***Coloca uma string numa variavel tipo apontador allocando tamanho automatico***/
char* Putstr(char* str);
/***Lê stdin e aloca automaticamente espaço em memória***/
char* ReadConsole(FILE *stream, size_t n);
/***lê apenas numeros de uma string***/
int getnum(int min, int max);
/***preenche vector de tamanho size por num***/
int* fillvec(int num, int size);
/***ReadFiletoMem***/
void* ReadFiletoMem(void* datatype,FILE* filename);
/***GetChar***/
unsigned char GetChar();
/***ReadConsoleR***/
char* ReadConsoleSer(FILE* stream);
/***gets the number out of a string***/
int getnumber(char* x);
/*******/
```

```
int infor(char* inf, int Size_Inf, FILE* stream);
/*******/
int SetupSerial(int fd,struct termios *oldtio,struct termios *newtio,speed_t speed);
/*******/
int ReadInt(int nmin, int nmax);
/*******/
int Write(int fd,char *string,char *end);
/*******/
char *MOME(FILE *fp,char keygen[2][word_size],char input[word_size]);
/*******/
#endif
#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
```

```
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
```

```
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
```

```
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
```

```
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
```

```
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
```

```
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
```

```
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
```

```
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
```

```
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1
```

```
int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
```

```
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
```

```
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;
```

```
  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
```

```
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(100);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(30);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(30);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/************************************************************/
```

```
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin
```

```
void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
```

```
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
```

```
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
```

```
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;
```

```
  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){

    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
```

```
    }
  }
  return 0;
}
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
#define BufSize 127
#define C 0
#define B 1

int getnum(char *x);
int SerialRead(char *state);

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
  DDRC=B11000000;
  PORTC=B00111111;
  DDRB=B00111111;
  PORTB=B00000000;

}
//begin

void loop()
{

  uint8_t Entry[2];
  uint8_t Hist[2];
  char State[BufSize];
  uint8_t flag;
  Hist[C]=0;


  for(;TRUE;Hist[C]=Entry[C],Hist[B]=Entry[B]){
```

```
    Entry[C]=PINC;
    delay(10);
    if(Serial.available()){
      SerialRead(State);
      Entry[B] = getnum(State);
    }
    delay(10);
    /*****************************************************/
    if(Entry[C] != Hist[C])
      Serial.println(Entry[C],DEC);
    delay(10);
    if(Entry[B] != Hist[B])
      PORTB = Entry[B];
    delay(10);
  }
}
//have to press reset in learning mode always.
/***FUNCTIONS***/
int getnum(char* x)
{
  int num;
  if(sscanf(x,"%d",&num)){
    if (num == NULL)
      num = 0;
    return num;

  }else{
    return 0;

  }

}
/*************************************************************/
int SerialRead(char *State)
{
  uint8_t i=0;
  char IncomingByte;
  delay(60);//wait for incoming data.
  for(i = 0; IncomingByte = Serial.read(); i++){
    if((IncomingByte == '\r') || (IncomingByte == '\n')){
      State[i] = '\0';
      Serial.flush();
      break;
    }else{
      State[i]=IncomingByte;
    }
  }
  return 0;
}
```

```
//char *X, X is a variable that stores a fisical
//address with a cast type char.
//getnum works with string terminating with only NULL or '\0'
//my style
//Lesson always be flexible and except and obey the good practices,
// without any desobidience, and for extra culture if desired try to
// clarify the why things are as they are. Never be stuborn.
//must flush buffer in the PC side.

CC=gcc
LIB=-L./


all:resultado

resultado:creation_7.o creation_7_func.o
	${CC} creation_7.o creation_7_func.o -Wall -lm -o FSM.exe ${LIB}

resultado.o:creation_7.c
	${CC} -c creation_7.c -Wall -lm -o creation_7.o ${LIB}

resultado_func.o:creation_7_func.c
	${CC} -c creation_7_func.c -Wall -lm -o creation_7_func.o ${LIB}

clean:
	rm creation_7.o creation_7_func.o
```