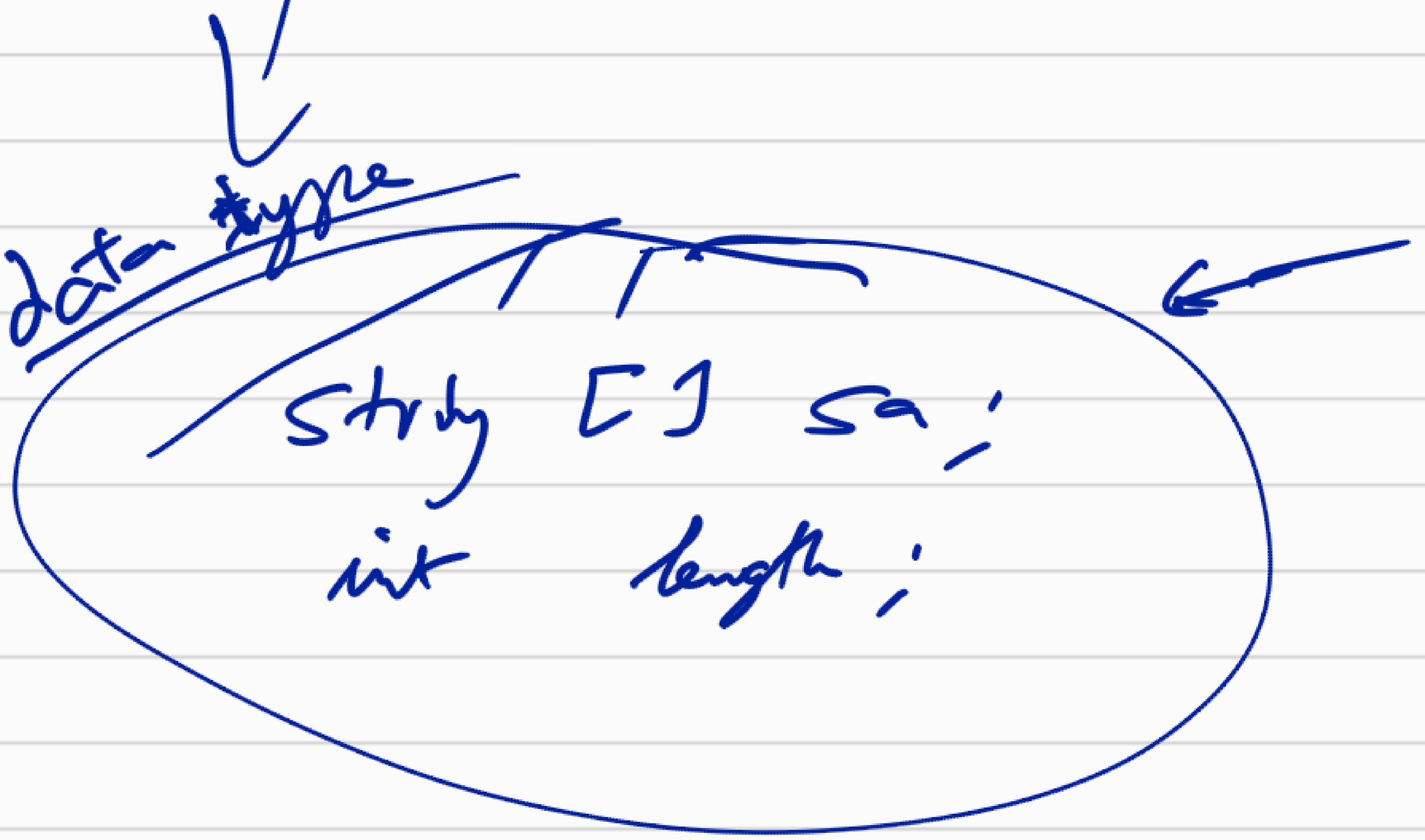
Data Structures :-

Goal is to come up with a "data structure" which stores strings along with their lengths.

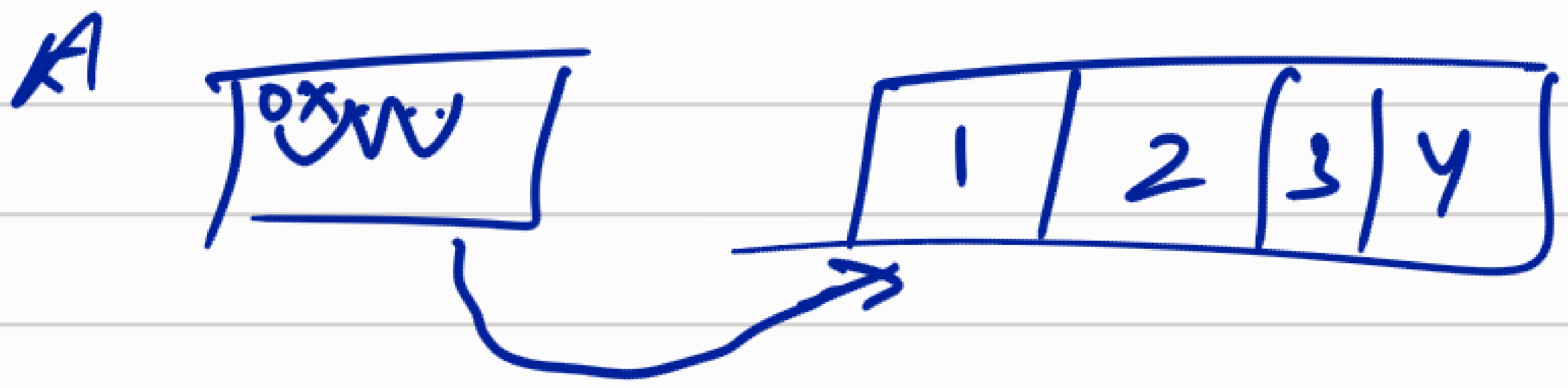
```
→ string [] sa ; -
→ int len-sa = 10 ;
→ sa = alloc - array ( string, len-sa) ;
```

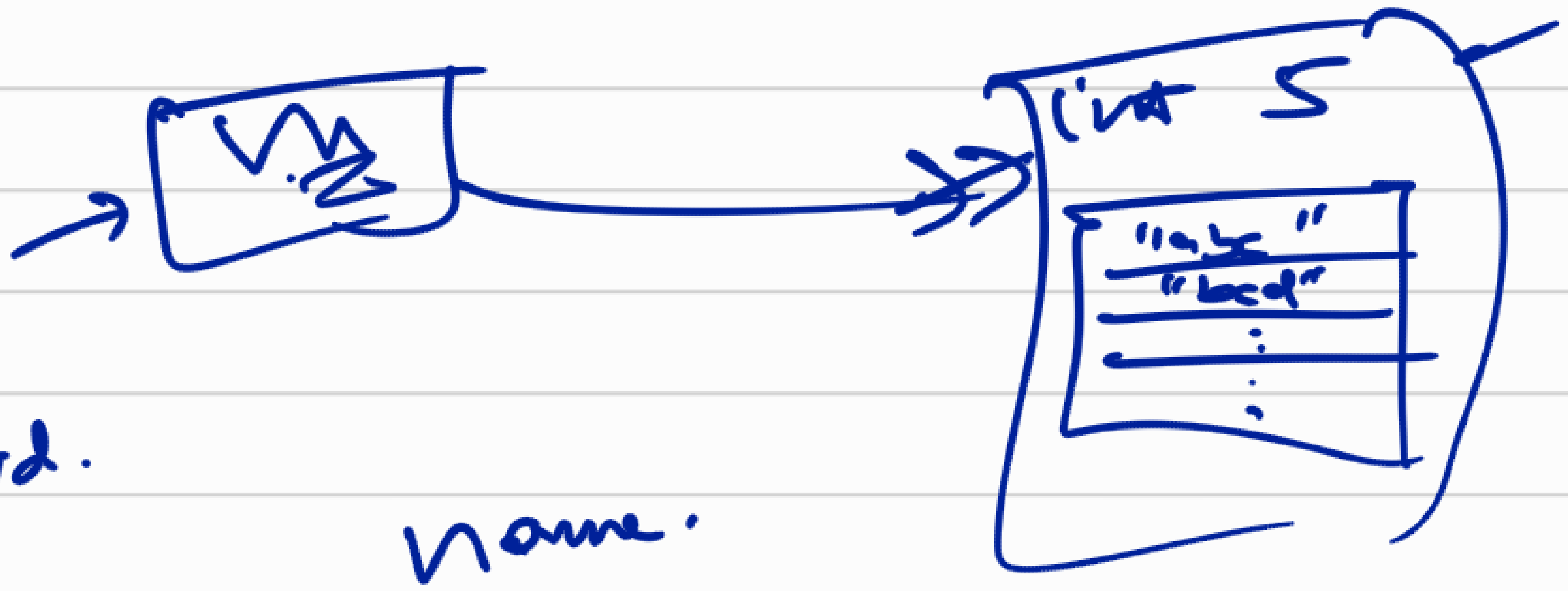


'int' 32 ← 32 bits in C.

int[] A;

local memory allocated.





keyword.

name.

```
struct ssa_header {
        int length;
        string [] data;
};
```

typedef struct ssa_header ssa;

typedef ssa* ssa_t;

```
/* interface */

typedef ssa* ssa-t; ✓

ssa_t ssa-new(int size)
  /*@ requires 0 <= size ; @*/
  /*@ensures \result != NULL ; @*/
  /*@ ensures ssa-len(\result)==size ; @*/;

int ssa-len ( ssa-t A)
  /*@ requires A != NULL ;     @*/
  /*@ ensures \result >= 0;     @*/ ;

string ssa-get (ssa_t A, int i)
 /*@ requires A!=NULL ;   @ */
 /*@ requires 0<= i && i < ssa-len(A);
                                  @*/ ;
// set ith string in A to x
void ssa-set (ssa_t A, int i, string x)

/*@ requires A!=NULL ;        */
/*@ requires 0<= i && i < ssa-len(A);
                                  @*/ ;
```

// implementation

```
string ssa_get (ssa* A, int i)
 /*@ requires  A != NULL;  @*/
 /*@ requires  0 <= i && i < ssa_len(A);
                                   @*/
{
    return   A -> data[i] ;
}


ssa*   ssa_new( int size )
  /*@ requires    0 <= size;  @*/
  /*@ensures   \result  != NULL ; @*/
  /*@ ensures   ssa_len (\result)==size ; @*/
{
    ssa*  A  =   alloc (ssa);
    A -> data =  alloc_array (string, size);
    A -> length =  size;
    return A;
}
```

```
int ssa_len (ssa* A)
/*@ requires A != NULL;  @*/
/*@ ensures \result >= 0;  @*/
{
    return A->length;
}
```

the problem is we dont really know if length is = \length(A->data)

```
void ssa_set(ssa* A, int i, string x)
{   Contracts
}
    A->data[i] = x;
```

data = ["a" , "b" ; "d"]; length=3

Self - sorting arrays:

① a string array: data

② its length : length

③ make sure that data is sorted & that length(data) = length.