COMPUTER ENGINEERING 12
PROJECT 3

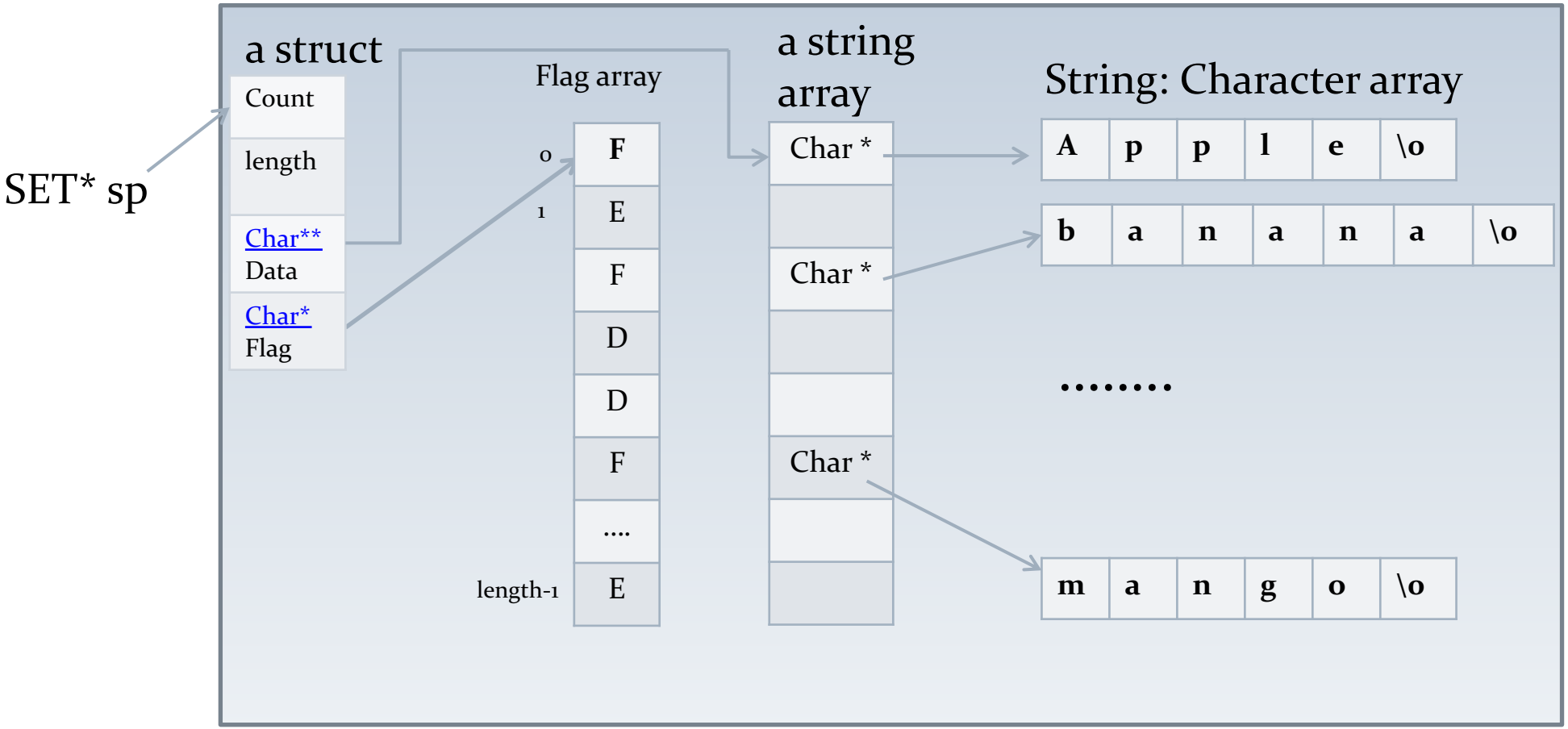
Xiang Li
xli8@scu.edu

PROJECT 3 – IMPLEMENTING A SET THROUGH HASH TABLE

Lab 3 (Week 1)

Using hash table to build a set.

The Data Structure (Part 1)



How to map the strings (i.e. keys) to the array addresses?

A Hash Function!

The Hash Function Adopted

```
unsigned strhash(char *s) {
    unsigned hash = 0;

    while (*s != '\0')
        hash = 31 * hash + *s ++;

    return hash;
}
```

Recall getElements in Project 2

```
char **getElements(SET *sp)
{
char **elts;


assert(sp != NULL);

elts = malloc(sizeof(char *) * sp->count);
assert(elts != NULL);

return memcpy(elts, sp->data, sizeof(char *) * sp-
>count);
}
```

Project 3 – Part 2

Making it general !

Write an ADT that works on generic pointer types - void *

So that our ADT SET can store strings, pointers to structures, or whatever we like.

Good News!

Functions need no changes:

numElements

Change char * to void *

```
void * findElement (SET *sp, void *elt)
void * removeElement(SET *sp, void *elt)
void * addElement(SET *sp, void *elt)
void ** getElements(SET *sp)
```

However ...

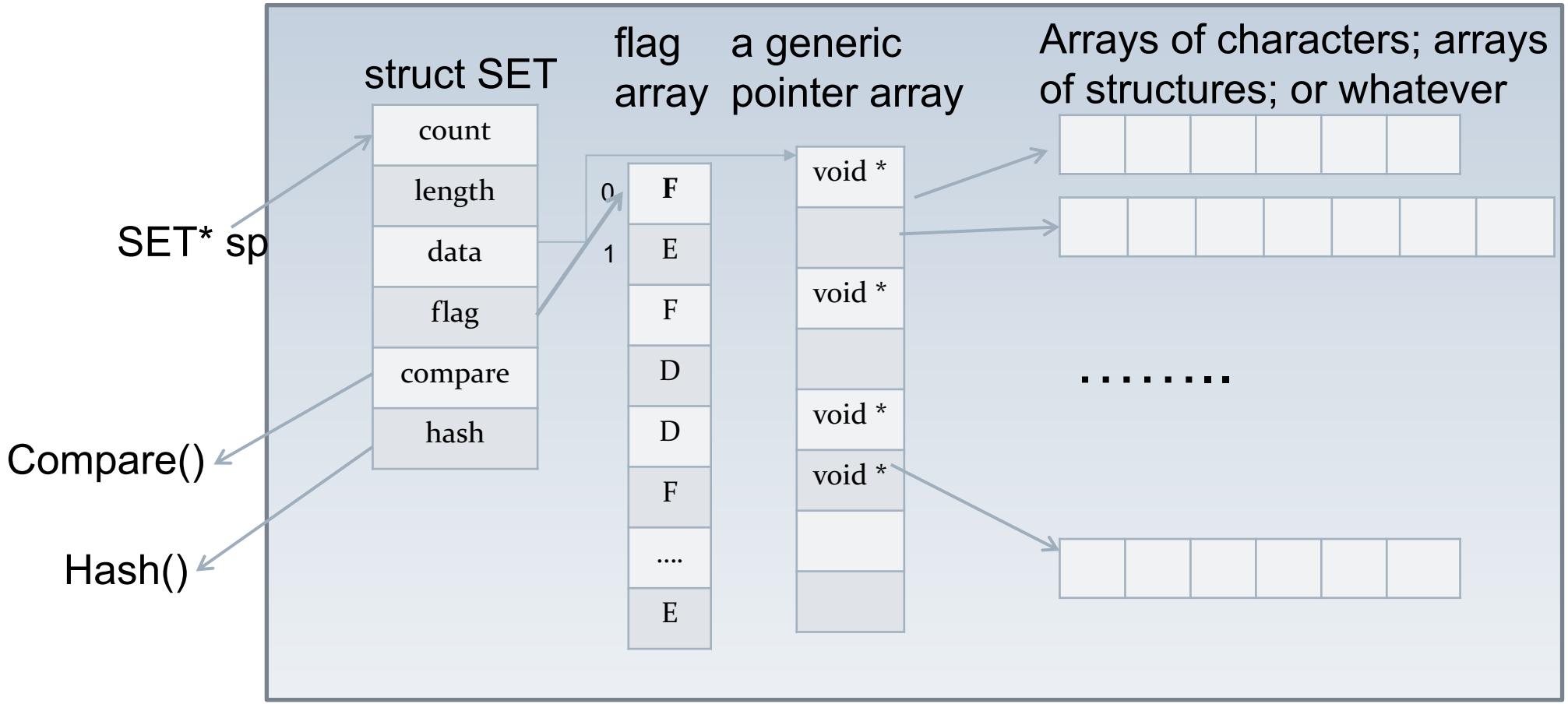
We need to do a little more than just replace “char *” with “void *” when we care the content/value of a specific element.
e.g. when we compare two elements; when we calculate the hash value of an element.

But as our ADT does not have knowledge about the specific type of each element, it’s the outsider program’s responsibility to implement such functions.

Our job is to take these functions as input parameters, and use them.

Please be aware that we only deallocate memory that is allocated by us. e.g. for destroySet function, we don’t need to free memory for each individual element.

The Data Structure of a SET



Function Pointers

The functions “compare” and “hash” will be passed to us through Function pointers.

```
struct set {
        int count;
        int length;
        void **data;
        char *flags;
        int (*compare)();
        unsigned (*hash)();
};
```

```
SET *createSet(int maxElts, int (*compare)(), unsigned (*hash)());
static int search(SET *sp, void *elt, bool *found)
```

When you call it, using (*sp->compare)(…)