2510 : Lecture 22

March 20, 2017

qsort

```
#include <stdlib.h>
int a[100];          size of each element
qsort (a, 100, sizeof(a[0]), cmp);
array name   # of elements      function pointer
                                comparison fcn
                                used to specify sorting order
```

prototype of cmp:

```
int cmp (const void *, const void *);
```

↳ when qsort needs to compare 2 elements, it passes in their addresses to cmp, it should:

1) return ∅ : if it doesn't matter which of the two elements go first

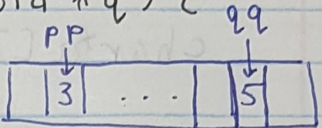
2) return a negative int # : if the element pointed to by p (ie. *p) should go <u>before</u> element pointed to by q (ie. *q) after sorting

3) return a positive int # : if the element point to by p should go <u>after</u> the elemente pointed to by q after sorting

... Lecture 22||

▷ sorting the integer array in descending order

```
int cmp(const void *p, const void *q) {
    const int *pp = p;
    const int *qq = q;
    return *qq - *pp;
}
```



ex. using struct

```
typedef struct {
    char id[10];
    int score;
} Score;
Score a[100]; /* assume we have stored 100 scores in array */
qsort(a, 100, sizeof(a[0]), cmp);
```

sort in 2 different ways:

① ascending order of IDs

```
int cmp(const void *p, const void *q){
    const Score *pp = p;
    const Score *qq = q;
    return strcmp(pp->id, qq->id);
}
```

(*pp).id ≡ pp->id

[if you have a pointer to structure, then arrow]

② descending order of scores, but if 2 or more Score structures have the same score, they are then sorted in ascending order of their IDs

```
int cmp(const void *p, const void *q){
    const Score *pp = p;
    const Score *qq = q;
    int n = qq->score - pp->score
    if (n != ∅)
        return n;
    return strcmp(pp->id, qq->id);
}
```

final exam material

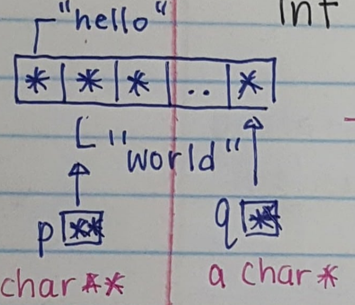
... Lecture 22 ||

Ex. sorting an array of strings

```
char *a [] = {"hello", "world", "goodbye", ... };
```

Sort in ascending order

```
qsort (a, sizeof(a)/sizeof(a[0]), sizeof(a[0]), cmp)
```



```
int cmp (const void *p, const void *q){
    char * const *pp = p;
```

→ we can't use p to change the pointer in the array

```
    char * const *qq = q;
    return strcmp(*pp, *qq);
```

✗ `const char ** pp = p;` compiler warning

✓ `const char ** pp = (const char **) p;` no warning

or

`char ** p = (char **) p;` no const

- what is the type of the 4th parameter of qsort?

```
void qsort (void *, size_t, size_t, ???);
```